Acquisition de grammaires lexicalisées pour les langues naturelles

Erwan Moreau

École Centrale de Nantes **Université de Nantes** **École des Mines de Nantes**

ÉCOLE DOCTORALE STIM

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DES MATÉRIAUX »

Année 2006

Acquisition de grammaires lexicalisées pour les langues naturelles

THÈSE DE DOCTORAT

Discipline : INFORMATIQUE

Présentée
et soutenue publiquement par

Erwan MOREAU

Le 18 octobre 2006 à l'UFR Sciences et Techniques, Université de Nantes,
devant le jury ci-dessous

Président	:	Pr. Béatrice DAILLE	Université de Nantes
Rapporteurs	:	Makoto KANAZAWA, Professeur	National Institute of Informatics, Tokyo
		Jean-Yves MARION, Professeur	Université de Nancy
Examinateurs	:	Alexandre DIKOVSKY, Professeur	Université de Nantes
		Annie FORET, Maître de Conférences *(invitée)*	Université Rennes 1
		Christian RETORÉ, Professeur	Université Bordeaux 1
		Isabelle TELLIER, Maître de Conférences H.D.R.	Université Lille 3

Directeur de thèse : Pr. Alexandre DIKOVSKY
Co-encadrant : Pr. Christian RETORÉ
Laboratoire: **L**ABORATOIRE D'**I**NFORMATIQUE DE **N**ANTES **A**TLANTIQUE.
2, rue de la Houssinière, BP 92 208 – 44 322 Nantes, CEDEX 3.

N° ED 0366-270

ACQUISITION DE GRAMMAIRES LEXICALISÉES POUR LES LANGUES NATURELLES

Learning lexicalized grammars for natural languages

Erwan MOREAU

⋈

favet neptunus eunti

Université de Nantes

Erwan MOREAU

Acquisition de grammaires lexicalisées pour les langues naturelles

IV+viii+236 p.

Ce document a été préparé avec LaTeX2e et la classe `these-LINA` version v. 1.47 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-LINA` est disponible à l'adresse :

`http://login.irin.sciences.univ-nantes.fr/`

Impression : mathese.tex – 20/11/2006 – 11:10

Révision pour la classe : these-LINA.cls,v 1.47 2006/06/30 17:40:06 mancheron Exp

Résumé

L'inférence grammaticale désigne le problème qui consiste à découvrir les règles de formation des phrases d'un langage, c'est-à-dire une grammaire de celui-ci. Dans le modèle d'apprentissage de Gold, les exemples fournis sont constitués uniquement des phrases appartenant au langage. L'algorithme doit fournir une grammaire qui représente le langage énuméré. Les grammaires catégorielles sont l'un des nombreux formalismes existants pour représenter des langages. Kanazawa a montré que certaines sous-classes de ces grammaires sont apprenables, mais ses résultats ne sont pas applicables directement aux langues naturelles.

Sur le plan théorique, nous proposons de généraliser les résultats de Kanazawa à différents types de grammaires. Les grammaires combinatoires générales sont un modèle flexible permettant de définir des systèmes grammaticaux à base de règles de réécriture. Nous démontrons dans ce cadre que certaines classes de langages sont apprenables. Dans un souci de généralité maximale, nos résultats sont exprimés sous forme de critères sur les règles des systèmes grammaticaux considérés. Ces résultats sont appliqués à plusieurs formalismes relativement adaptés à la représentation des langues naturelles.

Nous abordons également le problème de la mise en œuvre de l'apprentissage sur des données réelles. En effet, les algorithmes existants capables d'apprendre des classes de langages intéressantes sont NP-complets. Afin de contourner cet obstacle, nous proposons un cadre d'apprentissage plus souple, l'apprentissage partiel : le contexte d'utilisation est modifié dans le but d'obtenir une complexité algorithmique plus réaliste. Nous testons cette approche sur des données de taille moyenne, et obtenons des résultats relativement encourageants.

Mots-clés : Apprentissage automatique, Inférence grammaticale, Modèle de Gold, Identification à la limite, Grammaires lexicalisées, Grammaires catégorielles, Langues naturelles.

Abstract

Grammatical inference consists in discovering the rules governing how sentences of a language are formed, that is a grammar of this language. In Gold's model of learning, examples given as input are only sentences belonging to the language. The algorithm must provide a grammar which represents the enumerated language. Categorial grammars are one of the numerous existing formalisms used to represent languages. Kanazawa has shown that some subclasses of these grammars are learnable, but his results do not apply directly to natural languages.

In a theoretical viewpoint, we propose to generalize Kanazawa's results to different kinds of grammars. General combinatory grammars are a flexible model that permits to define grammatical systems based on rewriting rules. In this framework, we show that some classes of languages are learnable. In order to be maximally general, our results are expressed in the form of criteria on the grammatical system rules. These results are applied to several formalisms which are quite well suited for the representation of natural languages.

We also address the problem of implementing learning algorithms with real data. Indeed, existing algorithms that are able to learn rich classes of languages are NP-complete. We propose a more flexible learning framework, called partial learning, to bypass this obstacle: the context in which learning takes place is modified, in order to obtain a more realistic algorithmic complexity. We test this approach with some average size data, and obtain quite encouraging results.

Keywords: Automatic learning, Grammatical inference, Gold's model, Identification in the limit, Lexicalized grammars, Categorial grammars, Natural languages.

Remerciements

Tout d'abord, je remercie mes encadrants Alexandre Dikovsky et Christian Retoré pour toute l'aide qu'ils m'ont apportée. Je remercie aussi Makoto Kanazawa et Jean-Yves Marion, mes rapporteurs, pour leur relecture attentive et la pertinence de leurs remarques. Et bien sûr je remercie Béatrice Daille, Annie Foret et Isabelle Tellier d'avoir accepté de participer à mon jury.

Je souhaite ensuite adresser mes remerciements les plus sincères à l'ensemble des personnes croisées au cours de ces années de thèse au LINA. Grâce à elles j'ai apprécié le cadre dans lequel j'ai travaillé, profitant de cette ambiance amicale qui favorise les échanges. Je leur dois une grande partie de ma motivation, et j'ai une grande estime pour tous ces enseignants et/ou chercheurs, débutants ou expérimentés, grâce auxquels j'ai découvert la richesse de ce métier.

Merci donc à tous les collègues que j'ai rencontrés, souvent bienveillants, encourageants, attentionnés, parmi lesquels Charlotte, Christian, Gerson, Jérémie, Pascal, et plein d'autres. Je suis plus particulièrement reconnaissant à ceux de la meilleure équipe de recherche du monde, Annie, Béatrice, Chantal, Emmanuel et les autres. Comme la recherche ne fonctionne pas seulement avec des chercheurs, je remercie également ces personnes qui nous aident au quotidien : Christine, François, Jean-Yves, etc.

J'ai aussi beaucoup appris au contact des personnes avec lesquelles j'ai enseigné, notamment Alain, Christophe, Didier, François, Frédéric, Guillaume, Louahab, René, et (même si je n'ai pas vraiment enseigné avec elle) Marie-Mad, qui reste ma directrice du Département d'Info préférée.

J'ai bien sûr une sympathie particulière pour mes camarades doctorant(e)s. J'ai beaucoup apprécié leur compagnie, notamment au sein de l'association Login[1]. En voici quelques-uns, cités au gré de ma mémoire parfois défaillante[2], que je remercie en particulier[3] :

Alban, avec lequel je n'avais aucune affinité a priori : comme quoi il ne faut pas se fier aux apparences ! Brice, Lamiaa, alors finalement on en est arrivés au bout ! Gwen, mon premier voisin de bureau et prédécesseur à la présidence de Login, qui m'a donné envie de contribuer à la vie associative des doctorants ; Benjamin, Nordine, Fabrice, loginiens dynamiques de la grande époque ; Lucas, dont l'insatiable curiosité scientifique m'a impressionné ; Guillaume, fondateur de la CJCN, un beau projet ; Guillaume aussi, mon vice-président sympa et efficace ; Estelle, fausse informaticienne mais vraie enseignante-chercheuse exemplaire ; Sandra, qui m'a succédé à la tête de Login (et a remis l'asso sur la pente ascendante !) ; Cédric, fidèle voisin de bureau qui a supporté toutes mes mauvaises humeurs

[1] L'association des jeunes chercheurs en informatique de Nantes.

[2] Pour ceux que j'aurais oubliés : pardon ! je vous remercie quand-même beaucoup aussi, et même plus encore !

[3] Par ordre vaguement chronologique d'apparition. Bien sûr, ces quelques mots maladroits ne sauraient exprimer correctement ma profonde reconnaissance envers eux.

(ça n'a pas dû être facile tous les jours !) ; Amenel, collègue discret de compagnie agréable ; Anthony (le président), incroyablement dynamique et efficace, mais aussi attentif et à l'écoute ; Anthony aussi, voisin de bureau qui se prétend insupportable pour masquer sa gentillesse ; Jorge, avec qui je n'ai pas appris l'Espagnol, mais j'ai bien apprécié nos conversations le soir au RU ; Lorraine, pas avare de son temps pour les autres ; Sarah, militante à l'enthousiasme débordant ; Thomas, ex-président exagérément modeste ; Benoît, dont j'envie l'immense culture ; Manu, jeune président énergique de Login (qui ambitionne de devenir maître du monde et qui serait bien fichu d'y parvenir) ; pour finir, une petite pensée aussi pour Christophe, Nicolas, Frédéric, ainsi que les (nombreux ?) autres "soldats inconnus tombés au champ d'honneur".

Je remercie Jeremy, avec qui j'ai bien aimé travailler (et surtout ne pas travailler !), sans qui je ne me serais peut-être pas autant intéressé à la recherche en général et au TALN en particulier. Il y a encore de nombreuses autres personnes sans qui je n'aurais pas fait cette thèse, mais l'énumération exhaustive en serait quelque peu difficile !

Je remercie ma famille, mes parents Anne-Marie et Maurice et mes sœurs Gwénaëlle et Anne-Gaëlle. Leur présence constante et bienveillante à mes côtés a sans doute été le soutien le plus important dont j'ai bénéficié (aussi bien avant que pendant la thèse).

Enfin je remercie mon grand-père Jean, pour son exemple de volonté, de courage et de persévérance.

Ce travail est dédié à la mémoire de Jean Fleureau, mon grand-père,
décédé mardi 14 novembre 2006 à l'âge de 86 ans.

Sommaire

Introduction

Le langage est l'une des aptitudes qui symbolisent le mieux l'intelligence humaine. Le phénomène par lequel les enfants apprennent à parler en est la clé de voûte. Il était donc inévitable que l'apprentissage du langage devienne l'une des problématiques les plus représentatives de ce qu'on appelle l'intelligence artificielle. Mais la machine est encore bien loin de pouvoir rivaliser avec l'esprit humain dans ce domaine. Dans la continuation de certains travaux de Buszkowski et Kanazawa, nous étudierons comment l'apprentissage automatique peut s'appliquer aux langues naturelles. Cet objectif comporte un volet théorique (qu'est-ce que le processus d'apprentissage, qu'est-ce que les langues naturelles et comment les représente-t-on ?) et un volet plus appliqué (concrètement, comment et jusqu'où la machine peut-elle apprendre une langue ?).

Dès les premiers pas de l'informatique, le traitement du langage humain par la machine a été considéré comme l'un des objectifs essentiels de cette nouvelle science. En effet, la faculté de communiquer à l'aide d'un langage est certainement l'un des aspects les plus représentatifs de l'intelligence humaine. Or le langage naturel, même s'il est très complexe, est un objet que l'on peut étudier, formaliser, représenter : autrement dit un objet d'étude scientifique, celui dont traite la linguistique. En tant qu'objet formel, on peut envisager sa représentation et sa manipulation à l'aide de processus automatiques. C'est là l'objet de la linguistique computationnelle (au sens large), dont les origines remontent aux années 40, avec les premiers projets de traduction automatique.

De fait, le terme de *linguistique computationnelle* (on parle aussi de *linguistique informatique*) peut prendre deux sens assez distincts : au sens large, il s'agit de l'ensemble des concepts et techniques touchant à la représentation et à la manipulation des langues par la machine. Cette définition inclut la conception et la réalisation concrète d'algorithmes de traitement des langues pour des applications variées. Le Traitement Automatique du Langage Naturel est le domaine dédié à cette mise en œuvre, qui peut prendre des formes très éloignées d'une quelconque réalité linguistique, en particulier si les contraintes techniques ou les résultats le justifient. Mais cette définition inclut également une seconde notion, qui correspond au second sens que l'on peut donner au terme de *linguistique computationnelle* : celle-ci est alors considérée comme une spécialisation informatique de la linguistique, dont l'objet est l'étude théorique de la langue. Dans cette optique, cette dernière est conçue comme un processus totalement formalisable, à l'aide d'outils issus notamment de théories mathématiques et linguistiques. L'objectif premier n'est alors pas *l'application* mais la *représentation* informatique du langage naturel, dans ses différents aspects.

L'un des aspects les plus étonnants du langage humain est la relative facilité avec laquelle un enfant est capable d'acquérir sa langue maternelle, malgré la grande complexité d'une telle tâche. Or la réalisation d'un tel processus par la machine est un problème extrêmement difficile. Ce problème de l'apprentissage automatique de grammaires consiste à découvrir les règles de formation des phrases d'un langage particulier. L'ensemble des travaux que nous présentons a pour objectif l'étude des possibilités d'applications de l'apprentissage automatique aux langues naturelles. Pour situer plus précisément le contexte dans lequel s'inscrivent ces travaux, on peut sommairement placer cette étude à l'intersection de trois grands domaines, tous liés (de près ou de loin) au champ de la linguistique informatique :

- *L'inférence Grammaticale (IG).* Il s'agit de l'étude formelle du processus d'apprentissage de grammaires ou de langages à partir de données. Ce domaine a de nombreuses applications, de la bioinformatique à la fouille de données. Cependant le cœur de ce domaine réside dans la question suivante : pour une classe de langages fixée, y a-t-il un moyen de trouver le langage qui correspond à un ensemble de données fourni ? Bien entendu il y a de nombreux facteurs à prendre en compte, parmi lesquels la nature des données (phrases, structures complexes, présence ou non d'informations négatives, etc.), le système formel utilisé pour représenter le fait d'apprendre (c'est-à-dire quel est le critère de succès ou d'échec de l'apprentissage, éventuellement la marge d'erreur autorisée), faisabilité pratique de cet apprentissage, etc. Dans cette étude nous n'utiliserons comme cadre formel du processus d'apprentissage que le modèle de GOLD d'identification à la limite, et ne considérons que le cas de données positives. Ce modèle impose de nombreuses contraintes, mais ne traite pas (a priori) de l'aspect algorithmique de l'apprentissage, et par conséquent n'impose aucune restriction concernant la rapidité du processus. En ce sens, on peut voir les preuves d'apprenabilité dans ce modèle comme des résultats d'apprenabilité théorique, et non pratique.

- *Les grammaires catégorielles et grammaires de dépendances.* Ces formalismes grammaticaux sont conçus pour modéliser les langues naturelles, avec en général la volonté de capturer avec précision les phénomènes linguistiques qui les caractérisent. Si l'on considère une définition très générale des grammaires de dépendances, incluant tout formalisme utilisant une relation de dépendance dans la représentation syntaxique, on peut y inclure les grammaires catégorielles. Néanmoins on peut aussi restreindre la définition autour de la notion de dépendance, proposée par TESNIÈRE [107] et développée dans la théorie Sens-Texte élaborée principalement par MEL'ČUK [66]. Du fait de leur intérêt au niveau de la représentation des langues naturelles, nous appliquerons certains résultats d'apprenabilité à quelques variantes ou sous-classes des grammaires catégorielles et de dépendances. D'un point de vue technique, les grammaires catégorielles ont de plus l'avantage d'avoir certaines propriétés favorables pour l'apprentissage.

- *Le Traitement Automatique du Langage Naturel (TALN)*, et plus spécifiquement *l'apprentissage automatique.* Le Traitement Automatique du Langage Naturel a pour objectif de concevoir des outils informatiques capables de manipuler des langues naturelles : analyse syntaxique et/ou sémantique, génération, extraction d'informations de différentes natures. Dans ce vaste champ de possibilités, nous nous intéressons plus spécifiquement à ce qui concerne l'apprentissage automatique, c'est-à-dire les moyens de déduire des connaissances à partir de grands volumes de données en langage naturel. Dans ce cadre, il est possible d'exploiter toute forme de connaissances dont

on dispose sur la langue utilisée, de manière à obtenir des applications robustes et efficaces. La dernière partie de nos travaux (sur l'apprentissage partiel) se rattache directement à cet aspect du Traitement Automatique du Langage Naturel. Plus généralement, il s'agit de la visée à long terme de l'ensemble de cette étude, à savoir la réalisation concrète d'algorithmes d'apprentissage des langues, capables de fonctionner sur de gros volumes de données textuelles.

Notre approche se situe donc à l'intersection de plusieurs domaines assez distincts, et parfois antagonistes. Dans un sens, elle se rattache à la branche de la linguistique computationnelle visant la représentation formelle des langues (plus précisément dans notre cas l'acquisition des langues) : nous utilisons un modèle formel du processus d'apprentissage à la fois très strict et relativement fidèle aux caractéristiques de l'acquisition humaine de la langue. De plus, la pertinence linguistique est le critère principal de choix des formalismes grammaticaux que nous considérons. Mais notre approche s'inscrit aussi dans l'objectif de parvenir à de véritables outils de traitement des langues, en l'occurrence des outils d'inférence grammaticale. Dans cette perspective, l'adéquation au modèle humain de l'acquisition du langage n'est qu'un critère secondaire. Par exemple, nous verrons que les techniques d'apprentissage que nous proposons fonctionnent par généralisations successives du langage, alors qu'il est acquis que l'enfant apprend au contraire par spécialisation. Ainsi, notre approche se distingue des travaux dont l'objectif est la modélisation de l'acquisition humaine (tels ceux de TELLIER [105]) par la nécessité de parvenir à des outils informatiques utiles au traitement des langues (notamment au niveau de la nature des données fournies à l'algorithme). Elle se distingue également de la plupart des travaux liés à l'extraction automatique de grammaires d'après corpus (tels ceux d'HOCKENMAIER[55] ou de MOOT [71]), au sens où nous utilisons un modèle d'apprentissage théorique contraignant qui constitue une garantie de précision de la grammaire obtenue (l'apprentissage étant ici entièrement symbolique et ne faisant usage d'aucune forme d'heuristique ou de traitement statistique). Le modèle de GOLD est avant tout un modèle théorique, et le critère de convergence sur lequel il repose ne semble pas vraiment approprié pour des applications de traitement automatique. Pourtant, à la différence de l'apprentissage automatique classique sur corpus, ce critère donne une direction générale au processus de l'apprentissage : en effet, l'existence d'un objectif (qu'on peut considérer comme idéal) définissant ce que doit être la grammaire apprise diffère de la simple extraction d'information syntaxique, dans laquelle le résultat est justifié uniquement a posteriori (selon une évaluation spécifique ou simplement par son utilité). Mais les obstacles à l'application du modèle de GOLD aux langues naturelles sont nombreux, aussi bien sur le plan théorique que pratique. En conséquence, nos travaux doivent être vus simplement comme un pas de plus dans cette direction, et non comme une réponse complète à cette problématique. Soulignons par ailleurs que nous ne prenons en compte ici que l'aspect strictement syntaxique des langues.

Les applications de l'apprentissage automatique de grammaires du langage naturel sont nombreuses, si toutefois on admet l'hypothèse quelque peu idéaliste selon laquelle il est possible de construire un algorithme d'apprentissage "parfait". Celui-ci serait donc capable de donner une grammaire précise d'un langage naturel, pourvu qu'on lui fournisse un nombre suffisant de phrases appartenant à celui-ci. Sous cette hypothèse, la première application (et la plus évidente) est l'analyse syntaxique, elle-même utilisée sous différentes formes dans de nombreux outils de traitement des langues. On pourrait alors envisager de construire assez facilement des analyseurs, y compris pour des langues pour lesquelles peu d'études

linguistiques existent. On peut également penser à coupler l'analyse et l'apprentissage, de façon à mieux prendre en compte la catégorie syntaxique des mots inconnus de l'analyseur. D'autres applications liées à l'analyse, telles que la correction orthographique et syntaxique, sont également susceptibles de bénéficier des apports de l'apprentissage automatique. Si l'on dispose aussi des moyens permettant de gérer l'aspect sémantique des langues, l'autre grande application de l'apprentissage est la génération (passage du sens d'un énoncé à sa réalisation dans une langue précise). Celle-ci est elle-même proche du problème de la traduction automatique, qui est bien sûr une application d'une très grande utilité.

Le modèle de GOLD

L'acquisition du langage par la machine requiert en premier lieu une formalisation du processus d'apprentissage, afin d'en définir les objectifs et les moyens. Il existe plusieurs modèles répondant à ce besoin, parmi lesquels le modèle d'*Identification à la Limite*, plus communément appelé du nom de son auteur *modèle de* GOLD. Dans ce modèle, l'apprentissage est réalisé uniquement à partir d'exemples de phrases appartenant au langage ainsi décrit, à l'instar de l'acquisition humaine de la langue. On dispose d'une énumération infinie des phrases du langage, dont il faut extraire les points communs de manière à parvenir à une reconstitution fidèle de ce langage, représenté sous forme d'une grammaire.

Le modèle d'apprentissage proposé par GOLD en 1967 dans [50] est très restrictif, c'est pourquoi les premiers résultats obtenus avec ce modèle ont été négatifs. Par exemple, GOLD a lui-même démontré dans son article introductif que la classe des langages réguliers ne peut pas être apprise dans ce modèle. Mais, après les premiers résultats positifs obtenus au début des années 1980 par ANGLUIN [4], un certain nombre de travaux ont contribué à montrer l'intérêt et la pertinence de ce modèle : certes les classes de langages considérées comme les plus simples ne sont pas apprenables, mais il s'avère que certaines classes relativement riches, transversales aux critères classiques de classement des langages tels la hiérarchie de CHOMSKY, le sont. De nombreux outils théoriques sont alors développés pour caractériser ce qui est apprenable et ce qui ne l'est pas dans le modèle de GOLD, notamment certaines conditions suffisantes pour l'apprenabilité comme les *ensembles révélateurs* d'ANGLUIN [5], l'*élasticité finie* de WRIGHT [109] [80], la *densité finie bornée* de SHINOHARA [95]. Cette exploration progressive du modèle aboutit à des résultats importants au niveau des langages formels : SHINOHARA démontre que, pour tout entier positif k, la classe des langages générés par des grammaires contextuelles d'au plus k règles est apprenable.

Dans la continuité de cette évolution, les travaux menés sur le modèle de GOLD portent de plus en plus sur des formalismes grammaticaux concrets, plutôt que sur des classes de langages définies par des critères assez abstraits liés à la problématique de l'apprentissage. En 1998, KANAZAWA s'intéresse ainsi à l'apprenabilité des grammaires catégorielles classiques, souvent appelées grammaires AB. L'algorithme RG, proposé en 1989 par BUSZKOWSKI [27], constitue la base de ses travaux : il s'agit d'un algorithme d'apprentissage de grammaires AB rigides (limitées à une seule catégorie syntaxique par mot) à partir de FA-structures (structures complexes plus riches en information que de simples phrases). KANAZAWA démontre d'une part la convergence de l'algorithme (c'est-à-dire sa correction dans le modèle de GOLD), et en tire d'autre part plusieurs extensions, dont la plus générale est l'apprenabilité des gram-

maires AB d'au plus k règles (par mot) à partir de simples phrases, pour tout entier k. Il utilise pour cela différentes techniques de preuve originales, basées sur la propriété d'élasticité finie des langages, dont il montre les avantages. Il propose également une généralisation de l'algorithme RG à des formalismes proches des grammaires AB, mais limitée au cas des langages de structures de grammaires rigides.

Les résultats de KANAZAWA constituent le principal point de départ de nos propres travaux : il s'agit non seulement d'apports significatifs pour l'apprenabilité dans le modèle de GOLD, mais surtout ils ouvrent la voie vers des applications de ce modèle aux langues naturelles. En effet, si les grammaires AB en elles-mêmes sont relativement pauvres sur le plan linguistique, la variété des formes de grammaires catégorielles offre au contraire un cadre pertinent pour la représentation des langues naturelles. Il apparaît donc naturel de s'interroger sur les extensions éventuelles de l'apprenabilité à des classes de langages plus riches et proches des grammaires catégorielles.

Formalismes grammaticaux pour les langues naturelles

L'objectif de l'apprentissage étant de fournir une description du langage présenté, la question du formalisme grammatical est centrale pour les langues naturelles : dans quel "langage" doit-on (ou peut-on) les exprimer ? Les modèles de grammaires sont nombreux, variés et disposent chacun d'avantages et d'inconvénients, que ce soit au niveau de la pertinence linguistique, de l'expressivité, de la facilité d'utilisation par la machine, etc. Plutôt que d'étudier l'apprenabilité de chaque formalisme grammatical indépendamment des autres, nous essayons dans cette étude d'apporter des réponses aussi générales que possibles. Toutefois il ne faut pas non plus rechercher un niveau de généralisation trop élevé, car alors l'application des résultats d'apprenabilité à des formalismes concrets devient plus difficile[4]. Il est donc important de trouver une façon de caractériser les classes de langages apprenables à un niveau intermédiaire, qui permette à la fois d'utiliser les résultats sur des formalismes pour lesquels ils n'ont pas nécessairement été prévus, et que cette utilisation reste relativement simple et directe.

C'est pourquoi nous proposons d'utiliser un cadre de représentation des formalismes grammaticaux nommé les grammaires combinatoires générales (GCG). Ce modèle reprend, en l'élargissant, le système de même nom proposé par KANAZAWA parmi les ouvertures possibles de ses travaux. Dans ce modèle, un système de grammaires est défini par un ensemble de règles universelles qui contrôlent la façon dont sont dérivées les phrases. Chaque grammaire d'un tel système est composée de règles locales (plus ou moins complexes) définissant la base du processus de dérivation, à l'aide de catégories. Bien entendu, ce modèle permet d'exprimer des formalismes comme les grammaires AB, dont les règles universelles sont :

[4] Par exemple, ANGLUIN a démontré qu'une classe de langages est apprenable si et seulement s'il est possible d'énumérer un ensemble révélateur pour chaque langage de cette classe [5]. Cette caractérisation de l'apprenabilité est évidemment maximale au niveau des classes de langages qu'elle englobe, puisqu'il s'agit d'une condition équivalente à l'apprenabilité (dans le modèle de GOLD). Mais l'application de cette condition à un formalisme grammatical particulier est ardue, car le critère n'est pas aisément vérifiable sur une classe de langages donnée.

$$\mathcal{R}_{AB} = \left\{ \begin{array}{lll} (R_{FA}) & A/B & B \to A\,; \\ (R_{BA}) & B & B\backslash A \to A \end{array} \right\}$$

Ce modèle ne permet de représenter que les formalismes grammaticaux qu'il est possible d'exprimer sous forme de règles de réécriture. Mais il permet tout de même de représenter des systèmes n'ayant que peu de rapport avec les grammaires catégorielles. À titre d'illustration, voici une façon de représenter les expressions régulières à l'aide de règles universelles (avec $\mathcal{O}_{re} = \{\ \bullet(2)\ ;\ ^*(1)\ ;\ \cup(2)\ \}$ l'ensemble des opérateurs classiques de concaténation, fermeture de Kleene et union) :

$$\mathcal{R}_{re} = \left\{ \begin{array}{l} A \quad B \ \to\ A \bullet B\,; \\ \varepsilon \ \to\ A^*\,; \\ A \quad A^* \ \to\ A^*\,; \\ A \ \to\ A \cup B\,; \\ A \ \to\ B \cup A \end{array} \right\}$$

Une grammaire dans cette famille est constituée d'une unique règle de la forme $E \triangleright s$, avec E un type formé avec les opérateurs de $\mathcal{O}_{re}$. Par exemple, $\{\ (a^* \bullet b^*) \bullet (c \cup a)) \cup c^* \mapsto s\ \}$ est une grammaire de cette famille, représentant le langage décrit par l'expression régulière $(a^*b^*(c \cup a)) \cup c^*$.

Il devient ainsi possible, dans ce contexte, d'exprimer des conditions suffisantes pour l'apprenabilité sous forme de contraintes que doivent respecter les règles universelles. Ceci permet de vérifier assez facilement si un formalisme donné remplit un critère d'apprenabilité, dès lors que celui-ci peut être exprimé dans le modèle des grammaires combinatoires générales.

Aspects théoriques de l'apprentissage des langues naturelles

Naturellement, ce modèle est également choisi pour certaines de ses propriétés favorables à l'apprenabilité, déjà démontrées par KANAZAWA : toute classe de GCG rigides[5] est apprenable à partir de structures dans le modèle de GOLD. Mais celui-ci laissait ouverte la question de l'élasticité finie éventuelle de certaines classes de grammaires combinatoires générales. Cette question est essentielle, car l'élasticité finie est la propriété qui a permis à KANAZAWA de passer, dans le cas des grammaires AB, de l'apprenabilité des grammaires rigides à partir de structures à l'apprenabilité des grammaires k-valuées à partir de phrases simples. Or pour les langues naturelles, les contraintes de rigidité des grammaires et de nécessité de structures complexes sont doublement rédhibitoires : l'expressivité des grammaires rigides est clairement insuffisante, et les structures (spécifiques au formalisme) dont l'algorithme d'apprentissage a besoin sont difficilement disponibles en pratique pour des applications sur des volumes de données importants (ce qui est indispensable dans le cas des langues naturelles).

Dans le cadre des grammaires combinatoires générales, nous cherchons donc à définir des critères garantissant non seulement l'apprenabilité mais également l'élasticité finie des classes de langages. Cette

[5] KANAZAWA ne l'a démontré que par rapport à sa définition des GCG, c'est-à-dire uniquement pour des grammaires algébriques et lexicalisées. Nous verrons que son résultat peut être aisément étendu à une définition plus large.

recherche plutôt théorique s'inscrit dans l'objectif de découvrir à la fois des moyens d'appliquer l'apprentissage aux langues naturelles, et aussi les frontières qui séparent les classes apprenables des autres. Dans cette optique, nous étudions également les obstacles à l'apprenabilité et/ou l'élasticité finie, notamment à travers des exemples de classes de langages non apprenables, aussi proches que possible de cette frontière. Tout au long de cette exploration, nous cherchons à répondre à la question posée en filigrane de toute cette étude : le modèle de GOLD est-il adapté ou non à l'apprentissage automatique des langues ? Il faut souligner que l'apprenabilité au sens de GOLD est un concept de l'ordre de la décidabilité : étant donné un langage parmi une classe fixée et une énumération des phrases de ce langage, est-il possible de fournir au bout d'un nombre fini d'exemples une grammaire représentant de façon certaine et définitive ce langage ? C'est la raison pour laquelle il est indispensable de montrer l'apprenabilité d'une classe de langages avant de tenter de construire un algorithme qui effectue cette tâche : ne pas passer par cette étape préliminaire serait en quelque sorte comme chercher un algorithme qui résout un problème dont on ignore s'il est décidable. Dans le même ordre d'idée, il est important de noter qu'un résultat d'apprenabilité n'a que peu de sens indépendamment de sa démonstration[6], puisque qu'on doit pouvoir en déduire une méthode algorithmique d'apprentissage.

Nous proposons dans les chapitres 5 et 6 deux résultats positifs d'apprenabilité pour les grammaires combinatoires générales, sous forme de critères portant principalement sur les règles universelles utilisées. Ces deux résultats sont obtenus selon des méthodes sensiblement différentes : la première, qui concerne les *grammaires à arguments bornés* et est inspirée des travaux de SHINOHARA, est assez peu contraignante par rapport aux forme de règles universelles mais impose en revanche une limite sur certaines parties des types définis dans les grammaires. La seconde porte sur les *grammaires par consommation d'arguments*. À l'instar des résultats de KANAZAWA, elle ne requiert pas de limitation particulière sur la forme des grammaires, mais ne s'applique qu'à un nombre réduit de formalismes (du fait de contraintes plus importantes sur le type de règles universelles). Au niveau des classes de langages apprenables, ces deux résultats sont transversaux, c'est-à-dire qu'il existe à la fois des classes incluses dans les deux, et d'autres dans un seul des deux. Nous illustrons leur intérêt à l'aide de quelques applications à différents formalismes grammaticaux. Ici le terme d'"application" ne doit toutefois pas être entendu comme la mise en œuvre concrète d'un algorithme sur des données réelles, mais seulement comme l'instanciation d'une propriété générale à un système de grammaires spécifique.

Aspects pratiques de l'apprentissage des langues naturelles

Dans le cadre du modèle de GOLD, le mécanisme d'apprentissage est entièrement *symbolique*, c'est-à-dire qu'il n'y a aucune approximation par rapport aux données fournies : en supposant qu'on dispose de données "parfaites" (ne comportant aucune erreur par rapport à la grammaire qu'elles sont censées représenter), le risque d'erreur est réduit à zéro. Cette précision a un prix : le bon fonctionnement de l'algorithme d'apprentissage doit satisfaire des contraintes très fortes. C'est pourquoi le passage à l'échelle

[6] C'est pourquoi nous donnons tout au long de notre étude (y compris dans la partie bibliographique) des preuves détaillées, en privilégiant autant que possible une forme algorithmique. Ainsi le lecteur intéressé est-il en mesure de retrouver l'algorithme d'apprentissage correspondant au résultat démontré.

des données textuelles réelles est extrêmement difficile pour de nombreuses raisons : d'abord, la complexité algorithmique des méthodes d'apprentissage que l'on peut déduire des preuves d'apprenabilité est la plupart du temps exponentielle. Ensuite, le principe de l'identification à la limite se marie mal avec les contraintes du monde réel : en général on ne dispose pas d'une énumération infinie du langage, et quand bien même ce serait le cas la seule certitude de parvenir à la grammaire cible, sans savoir en combien de temps, est un peu faible pour de véritables applications. Enfin, la disponibilité des ressources est un obstacle à différents niveaux : existence de données sous la forme voulue (dans le cas de structures complexes), et surtout existence de données exemptes de toute "imperfection" par rapport au langage.

Dans la dernière partie nous abordons cette problématique de la mise en œuvre pratique de l'apprentissage de langues naturelles. Comme il s'agit d'un problème très vaste, nous nous intéressons plus spécifiquement à une partie de celui-ci, à savoir celle qui concerne la nature des données nécessaires en entrée de l'algorithme. En effet, l'utilisation de structures complexes en entrée apporte une plus grande quantité d'informations à l'algorithme d'apprentissage, ce qui permet d'en améliorer considérablement l'efficacité. Mais de telles structures ne sont généralement pas disponibles pour de gros volumes de données. Nous proposons donc un compromis, basé sur la méthode de KANAZAWA, qui conserve les avantages de l'apprentissage symbolique tout en éliminant cette contrainte sur les structures. En contrepartie, on considérera qu'une partie de la grammaire est déjà connue, de manière à remplacer l'information apportée par les structures par celle apportée par la *grammaire initiale*. Cette hypothèse est relativement réaliste dans la perspective de l'application aux cas réels, du fait notamment de la lexicalisation totale des grammaires traitées et de la "loi de ZIPF" qui se vérifie sur les textes en langue naturelle (celle-ci stipule que l'ensemble des mots les plus fréquents d'un texte est constitué d'une faible proportion de l'ensemble des mots, et réciproquement). L'inconvénient étant que l'efficacité de l'apprentissage dépend désormais beaucoup de la grammaire initiale.

Cette approche nommée *apprentissage partiel* fera l'objet d'un premier prototype dont nous présentons les résultats. Pour ces essais nous utilisons le lexique des grammaires de liens pour l'anglais. Cette grammaire a été construite par SLEATOR et TEMPERLEY [106], et offre une description assez précise et complète (au niveau grammatical) de l'anglais. Pour l'application que nous souhaitons en faire, les grammaires de liens ont surtout l'avantage d'être totalement lexicalisées et relativement proches des grammaires catégorielles. Cependant, il est nécessaire d'utiliser un formalisme intermédiaire exprimé dans le modèle des grammaires combinatoires générales pour répondre aux spécifications de l'algorithme, ce qui engendre certaines limitations par rapport à la grammaire d'origine. L'expérimentation réalisée tend à montrer que notre approche est pertinente, même si de nombreux problèmes subsistent : l'efficacité de l'algorithme sur des données de grande taille, le choix de la grammaire cible parmi l'ensemble des solutions retournées, et surtout le problème de la rigidité des grammaires.

PARTIE I

Langues naturelles et inférence grammaticale : deux domaines éloignés

CHAPITRE 1

Langages, grammaires et formalismes pour les langues naturelles

AVANT D'ABORDER LA PROBLÉMATIQUE DE L'APPRENTISSAGE DE LANGAGES, IL FAUT BIEN SÛR SE DONNER LES MOYENS DE LES REPRÉSENTER. LES FORMALISMES GRAMMATICAUX SONT DES SYSTÈMES METTANT EN RELATION DES LANGAGES AVEC DES DESCRIPTIONS DE CEUX-CI, NOMMÉES GRAMMAIRES. UNE GRANDE VARIÉTÉ DE FORMALISMES EXISTENT, CHACUN AYANT DES CARACTÉRISTIQUES SPÉCIFIQUES LE RENDANT PLUS OU MOINS ADAPTÉ À TEL TYPE DE LANGAGES OU TEL TYPE DE TRAITEMENTS. CETTE QUESTION EST FONDAMENTALE POUR LES LANGUES NATURELLES : LA PERTINENCE LINGUISTIQUE ET LA POSSIBILITÉ DE TRAITEMENT AUTOMATIQUE VIENNENT ALORS S'AJOUTER AUX CRITÈRES QUI DISTINGUENT CES FORMALISMES.

1.1 Introduction

Dans ce chapitre nous présentons quelques bases indispensables à l'étude des langages en général et des langues naturelles en particulier. Ce point de départ sert d'une part à définir les langages en général, concept central dans la problématique spécifique de l'apprentissage développée dans les chapitres suivants, et d'autre part à cerner les particularités des langues naturelles à travers la présentation de quelques formalismes grammaticaux adaptés à leur représentation.

Nous considérons ici la notion de langage d'un point de vue formel uniquement : un langage est simplement défini comme un ensemble (potentiellement infini) de phrases, la phrase étant un objet qui peut prendre différentes formes. La nécessité de représenter formellement des langages nous amènera à définir les concepts de grammaires et de systèmes de grammaires. Comme nous le verrons dans le chapitre 2, ces notions sont primordiales pour les problèmes d'inférence grammaticale, parce que le problème de l'apprentissage ne peut se définir que par rapport à un ensemble de langages et non par rapport à un langage isolé.

Les formalismes grammaticaux utilisés en Traitement Automatique des Langues sont nombreux et variés. En conséquence, ceux que nous présentons dans ce chapitre ne donnent qu'un bref aperçu des possibilités dans ce domaine. En dehors des très classiques grammaires de constituants qui servent de

référence, nous nous attachons plus particulièrement à la description de formalismes susceptibles de posséder certaines propriétés favorables à l'apprenabilité, notamment la famille des grammaires catégorielles. Malgré la richesse de ce type de grammaires et les nombreuses variantes existantes, nous nous concentrerons principalement sur leurs aspects syntaxiques. La syntaxe est en effet le seul niveau de représentation auquel nous nous intéresserons dans cette étude consacrée à l'inférence grammaticale, même si bien sûr ni les langues naturelles ni les formalismes considérés ne se réduisent à cet aspect.

Avant d'aborder ces concepts essentiels, nous effectuons quelques rappels concernant des définitions de base et précisons les notations et la terminologie employées tout au long de cette étude. Dans cette partie nous expliquons notamment les mécanismes concernant les termes (ou types) que sont la substitution et l'unification. Ces opérations jouent en effet un grand rôle pour les grammaires catégorielles, et nous les utiliserons beaucoup dans les définitions et démonstrations liées aux algorithmes d'apprentissage.

1.2 Terminologie, définitions et notations

1.2.1 Préliminaires

1.2.1.1 Ensembles, séquences

Un *ensemble* est une collection finie ou infinie d'objets, sans notion d'ordre ou de nombre d'occurrences. Les opérations suivantes sont définies sur les ensembles :

- $x \in A$ désigne le fait que l'élément x appartient à l'ensemble A ;
- $A \cup B$ désigne *l'union* des ensembles A et B, c.-à-d. l'ensemble composé des éléments appartenant à A ou à B ;
- $A \cap B$ désigne *l'intersection* des ensembles A et B, c.-à-d. l'ensemble composé des éléments appartenant à A et à B ;
- $A \triangle B$ désigne la *différence symétrique* des ensemble A et B, c.-à-d. l'ensemble composé des éléments appartenant soit à A soit à B, mais pas aux deux ;
- la relation binaire $\subseteq$ désigne *l'inclusion* : $A \subseteq B$ signifie que tout élément de A est aussi élément de B ($\subset$ désigne l'inclusion stricte : $A \subset B$ ssi $A \subseteq B$ et $A \neq B$) ;
- $A \setminus B$ désigne la différence de l'ensemble A par rapport à l'ensemble B, autrement dit l'ensemble des éléments de A n'appartenant pas à B.
- $A \times B$ désigne le produit cartésien de l'ensemble A avec l'ensemble B, c'est-à-dire l'ensemble constitué de toutes les paires (a, b) avec $a \in A$ et $b \in B$. De même le produit cartésien $A_1 \times A_2 \times \cdots \times A_n$ désigne l'ensemble de tous n-uplets $(a_1, a_2, \ldots, a_n)$ avec $a_i \in A_i$ pour tout i.
- $|A|$ est le nombre d'éléments de l'ensemble A ;
- $\mathcal{P}(A)$ désigne *l'ensemble des parties* de A, c.-à-d. l'ensemble constitué de tous les sous-ensembles (finis ou infinis) de A ;
- $\emptyset$ désigne *l'ensemble vide* .

Un ensemble est noté $A = \{a_1, a_2, \ldots, a_n\}$. Un ensemble A est un *sous-ensemble* d'un ensemble B si $A \subseteq B$, et réciproquement A est un *sur-ensemble* de B si $A \supseteq B$. Deux ensembles sont *disjoints* si leur intersection est l'ensemble vide.

Un *partitionnement* d'un ensemble A est une collection d'ensembles disjoints telle que l'union de tous ces ensembles est A. Autrement dit, $E = \{A_1, A_2, \ldots, A_n\}$ est un partitionnement de A si

- $A_i \cap A_j \neq \emptyset \Rightarrow i = j$;
- $\bigcup_{1 \leq i \leq n} A_i = A$.

Tout ensemble A_i est l'une des *partitions* du partitionnement E. Soit k un entier : le partitionnement $E = \{A_1, A_2, \ldots, A_n\}$ (avec $|E| = n$) est un *k-partitionnement* si $n \leq k$.

Une *séquence* est un ensemble ordonné d'objets (finie ou infinie). Un même objet peut apparaître plusieurs fois dans une séquence. Une séquence est notée $\langle a_1, \ldots, a_n \rangle$. La *longueur* d'une séquence S, notée $|S|$, est le nombre d'éléments dans cette séquence. La *concaténation* de deux séquences S et S', notée $S \bullet S'$, est la séquence composée d'abord de tous les éléments de S puis de tous ceux de S' : $\langle a_1, \ldots, a_n \rangle \bullet \langle b_1, \ldots, b_m \rangle = \langle a_1, \ldots, a_n, b_1, \ldots, b_m \rangle$. Les *chaînes* sont des séquences de mots[1]. Le cas particulier de la *chaîne vide*, c'est-à-dire la chaîne de longueur nulle, est noté ε[2].

L'ensemble des entiers naturels est noté $\mathbb{N} = \{0, 1, 2, \ldots\}$, et l'ensemble des entiers strictement positifs est noté $\mathbb{N}^* = \mathbb{N} \setminus \{0\}$.

1.2.1.2 Relations

Une relation est un sous-ensemble d'un produit cartésien : $R \subseteq A_1 \times A_2 \times \cdots \times A_n$. Pour une relation binaire $R \in A \times B$ (sur le produit cartésien de deux ensembles), on note souvent $a\ R\ b$ au lieu de $(a, b) \in R$, avec $a \in A$ et $b \in B$. Une relation binaire sur un ensemble A est une relation sur $A \times A$.

Une relation binaire R sur A est dite

- *réflexive* si $a\ R\ a$ pour tout $a \in A$;
- *transitive* si $a\ R\ b$ et $b\ R\ c$ implique $a\ R\ c$ pour tout $a, b, c \in A$;
- *symétrique* si $a\ R\ b$ si et seulement si $b\ R\ a$ pour tout $a, b \in A$;
- *antisymétrique* si $a\ R\ b$ et $b\ R\ a$ implique $a = b$ pour tout $a, b \in A$;

Une relation binaire $\preccurlyeq$ sur un ensemble A est un *ordre partiel* si $\preccurlyeq$ est réflexive, antisymétrique et transitive.

Une relation binaire $\sim$ sur un ensemble A est une *relation d'équivalence* si $\sim$ est réflexive, symétrique et transitive. Si $\sim$ est une relation d'équivalence sur A, alors la *classe d'équivalence* (selon $\sim$) d'un élément $a \in A$ est l'ensemble des $b \in A$ tels que $b \sim a$. Si A' et A'' sont deux classes d'équivalences de A selon $\sim$ alors $A = A'$ ou A et A' sont disjoints. Par conséquent l'ensemble des classes d'équivalence de A selon $\sim$ définit un partitionnement de A. Réciproquement, tout partitionnement de A définit une relation d'équivalence : si E est un partitionnement de A, alors la relation d'équivalence $\sim$ définie par ce partitionnement est définie par $a \sim b$ si et seulement s'il existe une partition A dans E telle que $a \in A$ et $b \in A$.

[1] ou de symboles, voir partie 1.2.3.

[2] On trouve également dans la littérature la notation λ.

Définition 1.1 (Fermeture réflexive et transitive)**.** La *fermeture réflexive et transitive* d'une relation binaire R sur un ensemble A est la relation binaire R' sur A définie inductivement par

- $a\ R'\ a$ pour tout $a \in A$;
- $a\ R'\ b$ pour tous $a, b \in A$ tels que $a\ R\ b$;
- $a\ R'\ c$ pour tous $a, b, c \in A$ tels que $a\ R'\ b$ et $b\ R'\ c$.

De toute évidence une telle relation est par définition réflexive et transitive.

1.2.2 Substitutions et unification

1.2.2.1 Termes, substitutions

Un *terme* est la composition d'*opérateurs* et de *variables*. Chaque opérateur a une *arité* fixée, indiquant le nombre d'opérandes de l'opérateur. L'arité d'un opérateur peut être nulle : on parle alors de constante[3].

Définition 1.2 (Terme)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs et $\mathcal{V}$ un ensemble de variables. L'ensemble des $\mathcal{O}$-termes sur $\mathcal{V}$ est défini inductivement par

- $v \in \mathcal{V}$ est un $\mathcal{O}$-terme sur $\mathcal{V}$;
- si $f \in \mathcal{O}$ est un opérateur d'arité n et si $t_1, t_2 \ldots, t_n$ sont n $\mathcal{O}$-termes sur $\mathcal{V}$, alors $f(t_1, t_2 \ldots, t_n)$ est un $\mathcal{O}$-terme sur $\mathcal{V}$.

Notons que le cas des opérateurs d'arité nulle est un cas particulier du second point de cette définition : tout opérateur d'arité nulle est donc aussi un $\mathcal{O}$-terme. On parle de *terme atomique* lorsque le terme n'est pas décomposable (il ne peut donc s'agir que d'une variable ou d'un opérateur d'arité nulle), par opposition au *terme complexe*.

Définition 1.3 (Sous-terme)**.** Un terme t est un sous-terme d'un terme t' si et seulement si $t = t'$ ou $t' = f(t_1, \ldots, t_n)$ et il existe i, $1 \leq i \leq n$, tel que t est un sous-terme de t_i.

On parlera de sous-terme *strict* si en plus le sous-terme t est différent du terme t'. La taille et la hauteur d'un terme sont définis de manière classique :

Définition 1.4 (Taille, hauteur)**.** La *taille* d'un terme t, notée $\|t\|$, est définie de la façon suivante :

- si t est atomique ($t \in \mathcal{V}$ ou t est un opérateur d'arité nulle), alors $\|t\| = 1$;
- si $t = f(t_1, \ldots, t_n)$ alors $\|t\| = 1 + \sum_{i=1}^{n} \|t_i\|$.

La *hauteur* d'un terme t, notée $h(t)$, est définie de la façon suivante :

- si t est atomique ($t \in \mathcal{V}$ ou t est un opérateur d'arité nulle), alors $h(t) = 0$;
- si $t = f(t_1, \ldots, t_n)$ alors $h(t) = 1 + \max(\{\ h(t_i) \mid 1 \leq i \leq n\ \})$.

[3] Il faut souligner ici la distinction entre variables et constantes, d'autant qu'on trouve parfois des conventions différentes dans la littérature selon l'utilisation qui est faite de la notion de terme. Dans la suite nous considérerons toujours les constantes comme un cas particulier d'opérateur. La distinction entre variables et opérateurs d'arité nulle est très importante pour les substitutions, puisque ces deux types d'éléments s'y comportent différemment.

De même, le nombre d'occurrences d'un terme dans un autre est défini formellement conformément à ce qu'on en attend :

Définition 1.5 (Nombre d'occurrences)**.** Le nombre d'occurrences d'un terme u dans un terme t, noté $\#_u(t)$, est défini de la façon suivante :

- si $u = t$ alors $\#_u(t) = 1$;
- sinon $u \neq t$:
  - si t est atomique, alors $\#_u(t) = 0$;
  - si $t = f(t_1, \ldots, t_n)$, alors $\#_u(t) = \sum_{i=1}^{n} \#_u(t_i)$;

$Var(t)$ (respectivement $Var(\{t_1, t_2, \ldots, t_n\})$) est l'ensemble des variables utilisées dans le terme t (resp. utilisées dans au moins un des termes de l'ensemble $\{t_1, t_2, \ldots, t_n\}$). La notion de terme sera très utilisée par la suite. Dans le domaine des grammaires catégorielles, dont ils forment la base, on parlera de *types*.

Définition 1.6 (Substitution)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs. Soit $\mathcal{V}$ et $\mathcal{V}'$ deux ensembles finis de variables et $Tp(\mathcal{V})$ (resp. $Tp(\mathcal{V}')$) l'ensemble des $\mathcal{O}$-termes sur $\mathcal{V}$ (resp. $\mathcal{V}'$).

Une substitution est une fonction $\sigma : \mathcal{V} \mapsto Tp(\mathcal{V}')$ qui associe à toute variable $v \in \mathcal{V}$ un terme de $Tp(\mathcal{V}')$. Une substitution est généralisée à une fonction $\sigma : Tp(\mathcal{V}) \mapsto Tp(\mathcal{V}')$ de la façon suivante :

- $\sigma(t) = t$ pour tout opérateur $t \in \mathcal{O}$ d'arité nulle ;
- $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ pour tout opérateur $f \in \mathcal{O}$ et tout ensemble de types $t_1, \ldots, t_n \in Tp(\mathcal{V})$.

La composition de deux substitutions $\sigma : \mathcal{V} \mapsto Tp(\mathcal{V}')$ et $\tau : \mathcal{V}' \mapsto Tp(\mathcal{V}'')$ est définie de manière usuelle par $(\tau \circ \sigma)(x) = \tau(\sigma(x))$ pour tout $x \in \mathcal{V}$. Cette définition vérifie comme on s'y attend que $(\tau \circ \sigma)(t) = \tau(\sigma(t))$ pour tout $t \in Tp(\mathcal{V})$. La *substitution identité*, qui associe à toute variable v de l'ensemble de départ la même variable v dans l'ensemble d'arrivée, est notée Id.

Définition 1.7 (Renommage de variables)**.** Un *renommage de variables* est une substitution telle que toute variable de l'ensemble d'arrivée a pour antécédent une et une seule variable de l'ensemble de départ. En d'autre terme, toute substitution bijective est un renommage.

Ainsi on parlera par exemple de grammaires *identiques modulo un renommage [de variables]*, ce qui signifie qu'il existe une substitution bijective permettant de passer d'une grammaire à l'autre. On verra dans la suite que de telles grammaires se comportent de façon totalement identiques, c'est pourquoi il est inutile de prêter attention aux variantes par renommage[4].

1.2.2.2 Unification

Définition 1.8 (Substitution plus générale)**.** Une substitution σ_1 est dite *plus générale* qu'une substitution σ_2 s'il existe une substitution τ telle que $\sigma_2 = \tau \circ \sigma_1$.

[4] Le terme de "variantes lexicographiques" (*alphabetical variants*) est souvent utilisé dans la littérature.

Définition 1.9 (Unifieur)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs et t et t' deux $\mathcal{O}$-termes sur $\mathcal{V}$. Une substitution $\sigma : \mathcal{V} \mapsto Tp(\mathcal{V}')$ est un *unifieur* pour le couple t, t' si $\sigma(t) = \sigma(t')$.

Cette définition est étendue à un ensemble de termes $A = \{t_1, \ldots, t_n\}$ de la manière suivante : σ est un unifieur pour A si $\sigma(t) = \sigma(t')$ pour tout couple $t, t' \in A$.

Enfin σ est un unifieur pour une *famille d'ensembles de termes* $\mathcal{A} = \{A_1, \ldots, A_n\}$ si σ est un unifieur pour tout $A \in \mathcal{A}$.

Définition 1.10 (Unifieur le plus général)**.** Étant donné un ensemble de types A (resp. une famille d'ensemble de types $\mathcal{A}$), σ est *l'unifieur le plus général* de A (resp. de $\mathcal{A}$) si pour tout unifieur σ' de A (resp. de $\mathcal{A}$) σ est plus général que σ'.

La propriété suivante découle directement de cette définition et sera souvent utilisée par la suite : si σ_{mgu} est l'unifieur le plus général de A (resp. $\mathcal{A}$) et que σ est aussi un unifieur de A (resp. $\mathcal{A}$), alors il existe une substitution τ telle que $\sigma = \tau \circ \sigma_{mgu}$.

L'unifieur le plus général d'un ensemble de types A (resp. d'une famille d'ensembles de types $\mathcal{A}$), noté $mgu(A)$ (resp. $mgu(\mathcal{A})$), a les deux propriétés remarquables suivantes :

- L'unifieur le plus général est unique (à un renommage de variables près).
- Il existe un unifieur de A (resp. de $\mathcal{A}$) si et seulement si $mgu(A)$ (resp. $mgu(\mathcal{A})$) existe.

La première propriété est facile à observer : par définition, si σ et σ' sont chacun un unifieur le plus général de $\mathcal{A}$ alors il existe deux substitutions τ et τ' telles que $\sigma = \tau' \circ \sigma'$ et $\sigma' = \tau \circ \sigma$, ce qui implique $\sigma = \tau' \circ \tau \circ \sigma$ (et $\sigma' = \tau \circ \tau' \circ \sigma'$). Il est clair alors que $\tau \circ \tau' = \tau' \circ \tau = Id$, donc τ et τ' sont des renommages et par conséquent σ et σ' sont identiques modulo un renommage de variables.

La seconde propriété est l'objet d'un théorème de ROBINSON [92] :

Proposition 1.1 (Théorème d'unification)**.** *Il existe un algorithme (appelé algorithme d'unification) qui étant donné deux termes*

- *renvoie leur plus général unifieur s'il existe un unifieur ;*
- *retourne un échec dans le cas contraire.*

Le lecteur intéressé trouvera une démonstration de cette proposition (ainsi que de nombreuses précisions concernant l'unification) dans [9]. Nous donnons ci-dessous (algorithme 1.2) un algorithme d'unification très simple tiré de [63][5], fondé sur celui proposé par ROBINSON. Il en existe d'autres versions beaucoup plus efficaces (notamment de complexité quadratique en utilisant des structures plus appropriées, voir [63]).

L'unification sur un ensemble de types ou sur une famille d'ensemble de types s'effectue facilement à partir de cet algorithme. L'algorithme 1.3 en est une version simple.

[5] La version que nous proposons précise la partie nommée *unifier_substs* (algorithme 1.1, nommé *compose* dans [63]), car KNIGHT omet dans cet algorithme de détailler les difficultés liées à la propagation des substitutions. En effet, les substitutions issues de sous-termes différents peuvent définir deux fois la même variable, ou causer une unification infinie lorsqu'elles sont composées.

unifier_substs(σ_1,σ_2)
Soit $\sigma_{mgu} = \emptyset$
Pour chaque x tel que $\{x \mapsto t\} \in (\sigma_1 \cup \sigma_2)$ **faire**
Si ($\{x \mapsto t_1\} \in \sigma_1$ et $\{x \mapsto t_2\} \in \sigma_2$**) Alors**
Si ($\tau \leftarrow$ **MGU(**t_1,t_2**)** termine avec succès **et** x n'a aucune occurrence dans $\sigma_{mgu}(\tau(t_1))$**) Alors**
$\sigma_{mgu} = \sigma_{mgu} \cup \{x \mapsto \sigma_{mgu}(\tau(t_1))\}$
Sinon
Échec
Fin Si
Sinon
[Sinon il existe une seule substitution sur x (soit dans σ_1 soit dans σ_2)]
Si (x a une occurrence dans $\sigma_{mgu}(t)$**) Alors**
Échec
Sinon
$\sigma_{mgu} = \sigma_{mgu} \cup \{x \mapsto \sigma_{mgu}(t)\}$
Fin Si
Fin Si
Fin Pour

ALG. 1.1 – Algorithme d'unification de substitutions

MGU(t_1**,**t_2**)**
- **Si** (t_1 ou t_2 est une variable) **Alors**
  - Soit x cette variable et t l'autre terme
  - **Si** ($x = t$) **Alors**
    - **Renvoyer** $\emptyset$
  - **Sinon**
    - **Si** (x a une occurrence dans t) **Alors**
      - **Échec**
    - **Sinon**
      - **Renvoyer** $\{x \mapsto t\}$
    - **Fin Si**
  - **Fin Si**
- **Sinon**
  - Soit $t_1 = f(x_1, \ldots, x_n)$ et $t_2 = g(y_1, \ldots, y_m)$
  - **Si** ($f \neq g$ ou $m \neq n$) **Alors**
    - **Échec**
  - **Sinon**
    - Soit $\sigma \leftarrow \emptyset$
    - **Pour** i **de** 1 **à** n **faire**
      - **Si** ($\tau \leftarrow$ **MGU(**x_i**,**y_i**)** termine avec succès) **Alors**
        - $\sigma \leftarrow$ **unifier_substs(**σ**,** τ**)**
      - **Sinon**
        - **Échec**
      - **Fin Si**
    - **Fin Pour**
    - **Renvoyer** σ
  - **Fin Si**
- **Fin Si**

ALG. 1.2 – Algorithme d'unification

MGU_set(A**)**
Si ($|A| > 1$) **Alors**
Soient $t, t' \in A$
Si ($\sigma \leftarrow$**MGU(**t**,**t'**)** termine avec succès) **Alors**
Si ($\tau \leftarrow$**MGU_set(**$(A \setminus \{t, t'\}) \cup \{\sigma(t)\}$**)** termine avec succès) **Alors**
Renvoyer τ
Sinon
Échec
Fin Si
Sinon
Échec
Fin Si
Sinon
Renvoyer $\emptyset$
Fin Si

MGU($\mathcal{A}$**)**
Soit $\sigma_{mgu} = \emptyset$
Soit $\mathcal{A} = \{A_1, \ldots, A_n\}$
Pour i **de** 1 **à** n **faire**
Si ($\sigma \leftarrow$ **MGU_set(**A_i**)** termine avec succès **et** $\tau \leftarrow$ **unifier_substs(**σ**,**σ_{mgu}**)** termine avec succès) **Alors**
$\sigma_{mgu} \leftarrow \sigma$
Sinon
Échec
Fin Si
Fin Pour

ALG. 1.3 – Algorithme d'unification d'une famille d'ensemble de types

1.2.3 Terminologie

Avant d'aborder les définitions de quelques formalismes grammaticaux il est sans doute utile de préciser quelques notions très générales ainsi que les choix terminologiques qui les accompagnent.

La notion de *langage* est au cœur de la problématique de l'apprentissage. Globalement un langage est un simple ensemble. Il n'y a pas a priori de restriction sur la forme que peuvent prendre les éléments d'un langage : bien entendu le cas le plus répandu est celui des langages de *chaînes*, c'est-à-dire celui où les éléments du langage sont des séquences de *mots*, les mots étant généralement pris dans un *vocabulaire* fixé. On distingue ainsi les *phrases* du langage (sous-entendu "les phrases correctes appartenant à ce langage") des autres séquences de mots. Mais nous étudierons également des cas où les langages ne sont pas de simples séquences de chaînes, en particulier des langages de *structures*, ce terme recouvrant différents niveaux de complexité possibles.

De plus, dans le cadre de la problématique de l'apprentissage de langages, on verra que la notion de langage isolé ne présente strictement aucun intérêt, c'est pourquoi il sera beaucoup question de *classes de langages* : une classe de langages est simplement un ensemble de langages. Il s'agira le plus souvent d'un ensemble de langages ayant tous une propriété particulière en commun. Notons qu'on trouve parfois dans la littérature du domaine (par exemple dans [61]) le terme *famille* de langages plutôt que classe de langages. On parlera parfois de *sous-classe* ou de *super-classe*, qui sont respectivement à une classe de langages ce que sont un sous-ensemble ou un sur-ensemble à un ensemble.

En théorie des langages formels, le vocabulaire (généralement noté Σ) est souvent nommé *alphabet*. L'alphabet est composé de *symboles*, et dans la plupart des exemples proposés chaque symbole est constitué d'un unique caractère. Dans cette optique, les éléments d'un langage sont donc nommés les *mots* (il s'agit en effet de séquences finies de symboles). Cependant, comme nous nous intéressons ici plus particulièrement au langage naturel, nous utiliserons toujours la terminologie *vocabulaire/mot/phrase* plutôt que *alphabet/symbole/mot* (le sens du terme "mot" étant bien sûr différent dans ces deux versions). Il est clair que cette forme où l'unité de base est le mot est nettement plus appropriée linguistiquement pour parler de syntaxe des langues naturelles.

1.2.4 Calculabilité, décidabilité et langages

On considère une définition classique de *machine de* TURING, et on utilise la thèse de CHURCH-TURING pour définir la notion d'algorithme : un *algorithme* est une procédure effective pouvant être représentée par une machine de TURING. Le langage accepté par une machine de TURING est l'ensemble des mots (entrées) pour lesquels la machine termine dans un état final. Dans certains cas il sera précisé explicitement[6] que l'algorithme *termine*, ce qui signifie qu'il doit "rendre une réponse" en un temps fini (ce qui est équivalent à l'existence d'une machine de Turing totale).

Remarque : dans les définitions ci-dessous nous traitons des langages car ceux-ci sont l'objet principal des présents travaux, mais ces même définitions s'appliquent directement aux ensembles en général.

[6] Si le contexte permet de le deviner de façon évidente, cela ne sera pas précisé.

Définition 1.11 (Langages récursivement énumérables)**.** Un langage L sur un univers $\mathcal{U}$ est *récursivement énumérable* (on dit aussi *semi-décidable*) s'il existe une fonction calculable $f : \mathbb{N} \mapsto \mathcal{U}$ telle que

$$\bigcup_{n \in \mathbb{N}} f(n) = L$$

On dit que f *énumère* le langage L. L'existence d'une telle fonction est équivalente à l'existence d'une fonction f' telle que pour tout $x \in \mathcal{U}$ f' termine si et seulement si $x \in L$.

Alternativement, on peut dire qu'un langage est récursivement énumérable s'il existe une machine de TURING qui accepte exactement ce langage, ou encore qu'un langage est récursivement énumérable s'il existe un algorithme qui termine si et seulement si son entrée x vérifie $x \in L$.

Définition 1.12 (Langages récursifs)**.** Un langage L sur un univers $\mathcal{U}$ est *récursif* (on dit aussi *décidable*) s'il existe une fonction calculable f telle que pour tout $x \in \mathcal{U}$

$$\begin{cases} f(x) = 1 & \text{si } x \in L \\ f(x) = 0 & \text{si } x \notin L \end{cases}$$

Alternativement, un langage est récursif s'il existe un algorithme indiquant si oui ou non $x \in L$ et qui termine pour tout $x \in L$.

Par définition, le *problème de l'appartenance au langage* ($x \in L$?) est décidable pour tout langage récursif. On peut bien sûr montrer qu'il existe des langages non récursifs, mais aussi des langages non récursivement énumérables[7].

1.3 Formalismes grammaticaux

L'étude formelle de langages nécessite de toute évidence un moyen de décrire des langages, et ce d'autant plus que les langages présentant un intérêt (typiquement les langues naturelles) ont une structure qui obéit à des règles particulières. Il est important de souligner qu'il n'est question ici que de syntaxe des langages, et donc de règles syntaxiques permettant de les représenter. Une *grammaire* est la description d'un langage, mais encore faut-il savoir "dans quel langage" est exprimée cette grammaire, autrement dit comment l'interpréter. Il est donc nécessaire de définir quel est le *formalisme grammatical* utilisé, sachant que de ce formalisme dépendent *l'expressivité* (la "variété" des langages représentables sous cette forme), la pertinence linguistique (la capacité de décrire tout ou partie des langues naturelles et l'éventuelle justification linguistique de cette représentation) et l'utilisabilité des grammaires (décidabilité du problème de l'appartenance, existence d'algorithmes de traitement efficaces, propriétés éventuelles d'apprenabilité, etc.).

De nombreux formalismes grammaticaux existent, chacun ayant ses avantages et ses inconvénients. L'objet de nos travaux étant l'étude des propriétés liées à l'apprenabilité dans un cadre aussi général que possible, nous utiliserons plusieurs de ces formalismes. C'est pourquoi il est utile de se doter d'une notion

[7] par diagonalisation avec les machines de Turing (par exemple).

formelle définissant les caractéristiques mininimales d'un tel système, suffisamment générale pour traiter de n'importe quel formalisme, mais aussi de n'importe quelle forme d'éléments du langage. En effet, la représentation de phrases du langage naturel peut nécessiter d'inclure des informations de type varié (par exemple des informations d'ordre sémantique). Il faut par conséquent inclure plus de souplesse que la stricte définition de langages de chaînes, de manière à permettre la manipulation de structures diverses (parenthésages de constituants, autres informations syntaxiques ou sémantiques, etc.). C'est pourquoi un *système de grammaires*, défini ci-dessous, devra définir à la fois les *objets* considérés, le type de grammaires, ainsi que le mécanisme par lequel un ensemble de tels objets est associé à une grammaire.

Définition 1.13 (Système de grammaires)**.** Un système de grammaires est un triplet $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ dans lequel $\mathcal{U}$ est un ensemble d'objets[8], $\mathcal{G}$ un ensemble de grammaires et $\mathcal{M}$ une fonction qui associe à chaque grammaire de $\mathcal{G}$ un sous-ensemble de $\mathcal{U}$: $\mathcal{M}(G)$ est le *langage* représenté par la grammaire G.

Ainsi les formalismes grammaticaux seront définis par le type d'objets qu'ils manipulent, représentant les phrases du langage (l'univers $\mathcal{U}$), la forme des grammaires utilisées ($\mathcal{G}$) et surtout la façon de relier une grammaire à l'ensemble de phrases qu'elle représente, son langage (la fontion $\mathcal{M}$).

1.3.1 Grammaires de constituants

1.3.1.1 Définitions

En théorie des langages formels, le formalisme de référence est sans conteste celui des grammaires de constituants (*phrase structure grammars*, aussi appelées *grammaires syntagmatiques* ou plus simplement grammaires de CHOMSKY) proposé par CHOMSKY dans [29].

Définition 1.14 (Grammaire de constituants)**.** Une *grammaire de constituants* est un tuple $\langle \Sigma, N, S, \mapsto \rangle$, dans lequel

- Σ est le vocabulaire ;
- N est l'ensemble des symboles non-terminaux ;
- $S \in N$ est un symbole non-terminal particulier ;
- $\mapsto \subseteq (\Sigma \cup N)^+ \times (\Sigma \cup N)^*$ est une relation binaire décrivant les règles de la grammaire.

Définition 1.15 (Dérivation)**.** Soit $G = \langle \Sigma, N, S, \mapsto \rangle$ une grammaire. La relation de dérivation immédiate $\Rightarrow_G \ (\Sigma \cup N)^+ \times (\Sigma \cup N)^*$ est définie par :

$\alpha \Rightarrow_G \beta$ si et seulement s'il existe $\alpha_1, \alpha_2, \alpha_3, \gamma \in (\Sigma \cup N)^*$ tels que

- $\alpha = \alpha_1 \bullet \alpha_2 \bullet \alpha_3$;
- $\beta = \alpha_1 \bullet \gamma \bullet \alpha_3$;
- et $\alpha_2 \mapsto \gamma$.

On définit ensuite la relation de dérivation $\Rightarrow^*$ comme la fermeture réflexive et transitive de $\Rightarrow$.

[8] Le terme générique "objets" est utilisé pour désigner les éléments d'un langage sans s'attacher à la forme précise de ceux-ci (voir partie 2.2.1.1).

De manière informelle, une chaîne peut être réécrite si elle contient une sous-séquence correspondant exactement à la partie gauche d'une des règles de production définies par la relation $\mapsto$, et le résultat de la réécriture est la chaîne de départ dans laquelle cette sous-séquence est remplacée par la partie droite de la règle. La relation de dérivation $\Rightarrow^*$ sert à appliquer autant de fois que nécessaire un tel remplacement.

Le langage d'une grammaire G, noté $L(G)$, est alors défini comme l'ensemble des chaînes $\alpha \in \Sigma^*$ telles que $S \Rightarrow^* \alpha$. Une grammaire G est *faiblement équivalente* à une grammaire G' si $L(G) = L(G')$. Deux grammaires sont dites *fortement équivalentes* si elles génèrent les mêmes ensembles de structures[9] de dérivation (c'est-à-dire que non seulement toute phrase appartenant au langage d'une grammaire appartient aussi au langage de l'autre, mais en plus ces phrases doivent pouvoir être construites avec des structures de dérivation isomorphes).

1.3.1.2 La hiérarchie de CHOMSKY

Le niveau de complexité des langages est très variable. Les grammaires de constituants offrent un moyen assez simple de caractériser la complexité des langages, à travers des critères facilement vérifiables sur les grammaires qui les représentent. La *hiérarchie de* CHOMSKY décompose les langages/grammaires en quatre classes, ayant chacune des propriétés très utiles qui font de cette hiérarchie une référence à laquelle on mesure souvent les autres formalismes grammaticaux. Voici ces quatre classes :

Type	Nom	Conditions
0	Non restreintes	Aucune restriction
1	Contextuelles	Toute règle est de la forme $\alpha \mapsto \beta$, avec $\|\alpha\| \leq \|\beta\|$.
2	Algébriques	Toute règle est de la forme $A \mapsto \alpha$, avec $A \in N$.
3	Rationnelles	Toute règle est de la forme $A \mapsto aB$ ou $A \mapsto a$, avec $A, B \in N$ et $a \in \Sigma$.

Comme on peut le voir, ces conditions impliquent que la classe des grammaires/langages de type i est toujours incluse[10] dans celle des grammaires/langages de type $i-1$. Notons qu'il existe quelques variantes de ces définitions, notamment à propos de la chaîne vide ε qui est parfois interdite comme unique élément de la partie droite dès les grammaires de type 1[11] : ces différences sont peu importantes dans la mesure où les classes de langages sont équivalentes.

Les grammaires non restreintes ont la propriété suivante : un langage est de type 0 si et seulement s'il est récursivement énumérable. Pour les grammaires contextuelles, aussi appelées *grammaires sensibles au contexte* (*context sensitive grammars*), on a la propriété que tout langage de type 1 est récursif (mais la réciproque est fausse en général), et donc que le problème de l'appartenance est décidable pour ces

[9] Il sera question plus en détail de langages de structures dans le chapitre 4.

[10] Il s'agit d'ailleurs d'une inclusion stricte.

[11] À une exception près pour $S \mapsto \varepsilon$, qui est toujours autorisé afin de permettre que la chaîne vide appartienne au langage représenté.

grammaires[12]. Les grammaires algébriques, aussi appelées *grammaires hors-contexte* (*context free grammars*), sont connues pour être en plus analysables en temps polynomial (voir [97], algorithme d'Earley [45], CKY [31],[62],[110]).

Les grammaires algébriques sont assez centrales dans la hiérarchie de CHOMSKY, du fait de leur expressivité souvent suffisante et de leurs propriétés qui les rendent facilement utilisables en pratique. Elles sont à ce titre très étudiées et très utilisées. Il existe en particulier deux formes normales importantes pour ces grammaires :

Définition 1.16 (Formes normales des grammaires algébriques)**.**

Une grammaire algébrique $G = \langle \Sigma, N, S, \mapsto \rangle$ est en *forme normale de* CHOMSKY si toutes ses règles sont de la forme $A \mapsto BC$ ou $A \mapsto a$, avec $A, B, C \in N$ et $a \in \Sigma$.

Une grammaire algébrique $G = \langle \Sigma, N, S, \mapsto \rangle$ est en *forme normale de* GREIBACH si toutes ses règles sont de la forme $A \mapsto aA_1A_2 \dots A_n$, avec $A_1, A_2, \dots, A_n \in N$ et $a \in \Sigma$. Si $n \leq 2$, la grammaire est dite en *forme normale forte de* GREIBACH.

Il est démontré que pour toute grammaire algébrique il existe une grammaire (faiblement) équivalente en forme normale de CHOMSKY et une grammaire (faiblement) équivalente en forme normale de GREIBACH, et que chacune de ces transformations peut être réalisée en temps polynomial ([29], [51], [56]). La transformation en forme normale de CHOMSKY est relativement fidèle à la grammaire d'origine (les arbres ne sont pas strictement isomorphes mais leurs structures sont très proches), mais la transformation en forme normale de GREIBACH peut donner des dérivations de forme très différente de celles d'origine.

Les grammaires rationnelles, aussi appelées grammaires régulières, sont la plus petite classe de cette hiérarchie, et ne décrivent par conséquent que les langages les plus simples. De tels langages peuvent être décrits à l'aide *d'expressions régulières* (voir par exemple [56],[11]), en utilisant uniquement les opérateurs suivants :

- Concaténation : $L \bullet L' = \{\ w \bullet w' \mid w \in L \text{ et } w' \in L'\ \}$;
- Union : $L \cup L' = \{\ w \mid w \in L \text{ ou } w \in L'\ \}$;
- Fermeture de Kleene : $L^* = \{\varepsilon\} \cup L \cup \{\ w \bullet w' \mid w \in L^* \text{ et } w' \in L^*\ \}$ (en d'autres termes L^* est l'ensemble des phrases obtenues par concaténation de n phrases de L, avec $n \geq 0$).

À ces opérateurs on ajoute par commodité les deux opérateurs suivants :

- Option : $L^? = \{\varepsilon\} \cup L$;
- Fermeture $+$: $L^+ = L \cup \{\ w \bullet w' \mid w \in L^+ \text{ et } w' \in L^+\ \} = L^* \bullet L = L^* \setminus \{\varepsilon\}$.

On utilisera également la notation $L^i = L \bullet L^{i-1}$, avec $L^0 = \{\varepsilon\}$, pour tout $i \in \mathbb{N}$.

[12] Ceci n'est pas difficile à constater d'après la forme des règles : en effet on montre aisément que le nombre de chaînes de longueur inférieure ou égale à un entier n fixé générées par de telles règles est borné.

1.3.1.3 Les langues naturelles dans la hiérarchie de CHOMSKY

La question de savoir où se situent les langues naturelles dans la hiérarchie de CHOMSKY a été largement étudiée et discutée. La question principale porte sur le fait que la frontière des langages algébriques soit ou non franchie par les langues naturelles : il est désormais admis que les langues naturelles sont "presque algébriques", mais sortent néanmoins légèrement des langages de type 2 et par conséquent ne sont pas des langages algébriques au sens strict [54]. En effet certains phénomènes linguistiques ne sont pas représentables par des grammaires algébriques. Voici quelques-uns des principaux points qui illustrent les limites des grammaires algébriques pour la représentation de langues naturelles (ces exemples sont issus principalement de [87]). Nous utilisons ici la notion de *dépendance*, dont il sera question plus en détail dans la partie 1.3.3. de manière informelle, il s'agit simplement d'éléments de description de la structure d'une phrase, qui relient les mots (ou les constituants) de la phrase à l'aide de relations syntaxiques.

Marie que Jean **voit**.
Marie que Pierre pense que Jean **voit**.
Marie que Pierre pense que Paul pense que ... que Jean **voit**.

Dans ces trois phrases "Marie" est l'objet direct de "voit", il y a donc une dépendance entre les deux mots, mais la distance qui les sépare est arbitraire : on parle de *dépendances distantes*. Ce type de phrases est acceptable par une grammaire de type 2 (au niveau des chaînes) mais la structure algébrique engendrée par la dérivation ne permet pas de représenter de façon linguistiquement satisfaisante le lien entre "Marie" et "voit".

Les *dépendances non projectives* (ou *croisées*) constituent la seconde catégorie de problèmes par rapport aux grammaires algébriques. De nombreux cas existent, parmi lesquels

- La négation (en français) : "Marie **n'**aime **pas** Jean" (constituants discontinus).
- Les clitiques : "Jean **en** mange **un morceau**". Ici "en" est le complément de l'objet "morceau", mais en est séparé par le verbe "mange" lui-même lié au sujet "Jean".
- Accords multiples (*multiple agreement*) : "**Peter**, **Paul** and **Mary** could achieve marks **ten**, **seven** and **eight** in **Mathematics**, **Linguistics** and **English**, respectively." On se trouve ici dans le cas d'un langage de la forme $a^n b^n c^n$ avec $n > 2$, dont il est démontré qu'il n'appartient pas à la classe des langages algébriques.

Malgré ces inconvénients, les grammaires algébriques fournissent une base relativement complète pour une représentation imparfaite mais globalement assez satisfaisante des langues naturelles, ou du moins d'une grande partie de celles-ci. Citons notamment le cas des grammaires de liens (voir partie 1.3.3.4), un formalisme équivalent aux langages algébriques, dont les auteurs se sont servis pour modéliser la langue anglaise de façon assez précise.

1.3.2 Grammaires catégorielles

Les *grammaires catégorielles* (*categorial grammars*) recouvrent un ensemble de formalismes grammaticaux ayant en commun la notion de *catégories*, qu'on appelle également *types*. Le cœur de chaque forme de grammaire catégorielle est un ensemble d'axiomes (ou de *règles universelles*), qui régissent la manière dont les catégories peuvent se combiner entre elles. Dans un tel système, une grammaire est définie uniquement par un *lexique*, c'est-à-dire l'association d'un ou plusieurs types à chaque mot du vocabulaire. On dit que les grammaires sont *totalement lexicalisées*, car c'est uniquement le lexique qui contient toutes les caractéristiques de la grammaire (contrairement par exemple aux grammaires de constituants qui contiennent souvent des règles où n'apparaît aucun mot du vocabulaire Σ).

Historiquement[13] les grammaires catégorielles trouvent leur source dans les travaux d'AJDUKIEWICZ, qui a défini un calcul à base de fractions pour tester la correction de propositions de logique, en lien avec les langues naturelles [3]. Ces travaux ont laissé dans les grammaires catégorielles classiques, introduites dans [13], le principe d'un calcul sur des fractions que l'on réduit au fur et à mesure du processus de dérivation. La barre de fraction est restée, mais elle s'est dédoublée pour devenir orientée et ainsi permettre la gestion de l'ordre des mots :

$$\text{De} \quad \frac{A}{B} \quad B \rightarrow A \quad \text{on est passé à} \quad A/B \quad B \rightarrow A \quad \text{et} \quad B \quad B\backslash A \rightarrow A.$$

Ces deux règles forment le noyau de tous les systèmes de grammaires catégorielles. Nous présenterons ces systèmes dérivés dans les parties 1.3.2.2 et 1.3.2.3. Le lecteur intéressé trouvera de plus amples détails sur les grammaires catégorielles et les systèmes modernes qui s'y rattachent dans [1], [90].

1.3.2.1 Grammaires AB

Les grammaires catégorielles classiques, nommées aussi grammaires AB, ont été introduites dans [13]. Ces grammaires sont totalement lexicalisées : cela signifie qu'une grammaire est décrite uniquement par son lexique, le lexique étant l'association d'une ou plusieurs catégories à chaque mot du vocabulaire. Les règles utilisées dans les dérivations sont donc *universelles*. Ces règles sont :

$$A/B,\ B \rightarrow A \quad \textit{FA (Forward Application)}$$
$$B,\ B\backslash A \rightarrow A \quad \textit{BA (Backward Application)}$$

Les *catégories* (aussi nommées *types*) sont des termes utilisant les opérateurs binaires $/$ et $\backslash$. Intuitivement, une expression est de type A/B (resp. $B\backslash A$) si cette expression est de type A lorsqu'elle est suivie (resp. précédée) par une expression de type B. Une phrase est correcte s'il est possible d'associer à chaque mot l'une de ses catégories, de telle sorte que les règles universelles permettent de transformer cette séquence de catégories en la catégorie spéciale s.

Définition 1.17 (Grammaire AB)**.** Une *grammaire AB* est un tuple $G = \langle \Sigma, Pr, s, \triangleright \rangle$, dans lequel

- Σ est le vocabulaire ;

[13] Voir [28] pour plus de détails sur l'histoire des grammaires catégorielles.

- Pr est l'ensemble des types primitifs. À partir de cet ensemble on définit l'ensemble des types de G, noté Tp_G, comme l'ensemble des $\mathcal{O}_{AB}$-termes sur Pr ; avec $\mathcal{O}_{AB} = \{\ /, \backslash\}$;
- $s \in Pr$ est un type primitif particulier désignant les phrases correctes du langage ;
- $\triangleright \subseteq \Sigma \times Tp_G$ est la relation finie associant à chaque mot du vocabulaire Σ un ou plusieurs types de Tp_G.

Remarque : lorsque cela ne présente pas d'ambiguïté, on notera simplement Tp l'ensemble des types d'une grammaire G.

Intuitivement, les deux règles universelles FA et BA sont des motifs que l'on peut appliquer à une séquence de types pour effectuer un pas de dérivation. Formellement :

Définition 1.18 (Dérivation)**.** Soit $G = \langle \Sigma, Pr, s, \triangleright \rangle$ une grammaire AB. La relation de dérivation immédiate $\Rightarrow_G\ Tp^+ \times Tp^+$ est définie par :

$\alpha \Rightarrow_G \beta$ si et seulement s'il existe $\alpha_1, \alpha_3 \in Tp^*$ et $X, Y, Z \in Tp$ tels que
- $\alpha = \alpha_1 \bullet \langle X, Y \rangle \bullet \alpha_3$;
- $\beta = \alpha_1 \bullet \langle Z \rangle \bullet \alpha_3$;
- et il existe une substitution σ telle que l'une des conditions suivantes est vérifiée :
  - $\sigma(A/B) = X,\ \sigma(B) = Y$ et $\sigma(A) = Z$;
  - $\sigma(B) = X,\ \sigma(B\backslash A) = Y$ et $\sigma(A) = Z$;

On définit ensuite la relation de dérivation $\Rightarrow^*$ comme la fermeture réflexive et transitive de $\Rightarrow$. Le langage de la grammaire G, noté $L(G)$, est l'ensemble des phrases $w_1 \dots w_n$ telles que $w_i \triangleright t_i$ pour tout i et $t_1 \dots t_n \Rightarrow^* s$.

Exemple 1.1. Soit G la grammaire constituée du lexique suivant :
$\{$ *Pierre, Marie, Paul* : SN; *aime, déteste* : $(SN\backslash S)/SN$; *qui* : $(SN\backslash SN)/(SN\backslash S)$ $\}$.

La phrase « Pierre, qui aime Marie, déteste Paul » appartient au langage de cette grammaire, comme le montre la dérivation suivante :

$$
\begin{aligned}
& SN,\ (SN\backslash SN)/(SN\backslash S),\ (SN\backslash S)/SN,\ SN,\ (SN\backslash S)/SN,\ SN \\
\Rightarrow\ & SN,\ (SN\backslash SN)/(SN\backslash S),\ (SN\backslash S)/SN,\ SN,\ SN\backslash S \\
\Rightarrow\ & SN,\ (SN\backslash SN)/(SN\backslash S),\ SN\backslash S,\ SN\backslash S \\
\Rightarrow\ & SN,\ SN\backslash SN,\ SN\backslash S \\
\Rightarrow\ & SN,\ SN\backslash S \\
\Rightarrow\ & S
\end{aligned}
$$

Une grammaire AB est *rigide* si chaque mot n'est défini que par une seule catégorie. De même, une grammaire est dite *k-valuée* si chaque mot est défini par au plus k catégories.

On peut décrire une dérivation à l'aide d'un arbre de manière classique, en étiquetant les nœuds par la catégorie du constituant qu'ils représentent. De plus, la forme des règles permet aussi de représenter un arbre de dérivation en étiquetant les nœuds seulement par l'identifiant de la règle utilisée (FA ou BA). Une telle structure, dans laquelle les feuilles ne sont étiquetées que par un mot, est appellée *FA-structure* (pour *Functor-Argument structure*). Cette représention est unique pour un arbre de dérivation donné (voir figure 1.1). En revanche une FA-structure donnée peut représenter un nombre infini d'arbres

de dérivations : dans la figure 1.1, on peut par exemple remplacer tous les SN par des $(X_1/X_2)/.../X_n$ et la FA-structure reste identique.

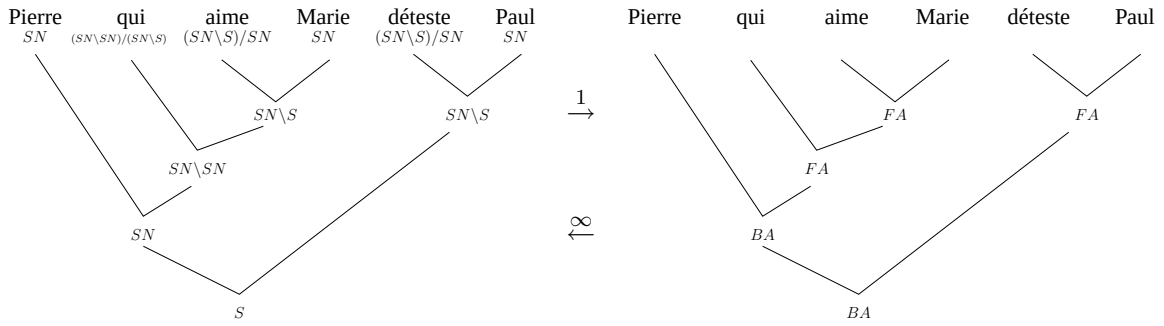


Figure 1.1 – Arbres de dérivations et FA-structure

Nous nous intéressons particulièrement à cette notion de FA-structure car celle-ci est à la base de l'algorithme d'apprentissage de grammaires AB défini par Buszkowski [27] puis enrichi par Kanazawa [60]. Ci-dessous sont définies plus formellement les notions de FA-structures, et de langages de structures et de chaînes :

Définition 1.19 (FA-structure)**.** Étant donné un vocabulaire Σ, l'ensemble des FA-structures $\mathcal{SL}_\Sigma$ est défini comme le plus petit ensemble tel que

- tout mot $w \in \Sigma$ appartient à $\mathcal{SL}_\Sigma$;
- pour tout couple $T_1, T_2 \in \mathcal{SL}_\Sigma$, les FA-structures $FA(T_1, T_2)$ et $BA(T_1, T_2)$ appartiennent à $\mathcal{SL}_\Sigma$.

Le *produit* d'une FA-structure est défini comme la séquence des mots qui apparaissent aux feuilles de celle-ci, c'est-à-dire la phrase représentée par cette structure.

Définition 1.20 (Instance d'une FA-structure)**.** Soit G une grammaire AB et T une FA-structure. Un couple (α, t_0), où α est une séquence non vide de catégories et t_0 une catégorie, est une instance de T pour G si l'une des conditions suivantes est remplie :

- si T est constituée uniquement d'un mot w, alors $\alpha = \langle t_0 \rangle$ et il existe une règle de G qui associe la catégorie t_0 à w ($w \triangleright t_0$) ;
- si $T = FA(T_1, T_2)$ (resp. $T = BA(T_1, T_2)$), alors il existe deux couples (α_1, t_1) et (α_2, t_2), respectivement instances de T_1 et T_2, tels que $t_1 t_2 \rightarrow_{FA} t_0$ (resp. $t_1 t_2 \rightarrow_{BA} t_0$) et $\alpha = \alpha_1 \bullet \alpha_2$.

Intuitivement un couple (α, t) est une instance d'une FA-structure T pour une grammaire donnée si le fait de "coller" la séquence de types α aux feuilles de la structure et le type t à la racine (et d'étiqueter les autres nœuds en conséquence) permet d'obtenir un arbre de dérivation partiel correct pour la grammaire. On peut ainsi définir le *langage de structures* d'une grammaire G, noté $FL(G)$, comme l'ensemble des FA-structures telles qu'il existe une instance (α, S) de T pour G. Le *langage de chaînes* $L(G)$ de la grammaire G est alors défini comme l'ensemble des chaînes x telles qu'il existe une structure $T \in FL(G)$ dont x est le produit. On vérifie aisément que cette définition est en tout point équivalente à celle de la définition 1.18.

La forme des règles laisse deviner que les grammaires AB sont algébriques. Il est en effet démontré dans [12] que toute grammaire AB est équivalente à une grammaire hors-contexte en forme normale de CHOMSKY. La transformation proposée est la suivante : l'ensemble des sous-types de tous les types du lexique est fini. On définit pour chaque sous-type un non-terminal, et pour chaque triplet de sous-types $\langle t, t', t'' \rangle$ dérivables par une des règles FA ou BA ($t, t' \Rightarrow t''$) on crée une règle $u'' \mapsto uu'$, où u, u', u'' sont les non-terminaux correspondant respectivement aux types t, t', t''. Pour chaque règle $w \triangleright t$ on ajoute aussi une règle $u \mapsto w$ (avec u le non-terminal correspondant au type t), de sorte que la grammaire algébrique ainsi construite a exactement le même langage que la grammaire AB d'origine. Il est également démontré dans [12] que toute grammaire hors-contexte (n'acceptant pas le mot vide ε) peut être convertie en grammaire AB[14], ce qui implique que la classe des langages des grammaires AB est exactement la classe des langages algébriques.

1.3.2.2 Grammaires de LAMBEK

On peut expliquer les développements ultérieurs des grammaires catégorielles de deux façons, toutes deux ayant pour but de pallier les limitations des grammaires AB (notamment en termes d'expressivité et de représentation pour les langues naturelles). Dans la première explication, ces limitations conduisent à l'ajout de règles supplémentaires aux règles universelles FA et BA dans le but de mieux représenter la complexité des langues naturelles. Dans la seconde explication, les règles universelles FA et BA sont réinterprétées en tant qu'axiomes d'un système logique, ce qui conduit à compléter le système de façon à le rendre plus cohérent mathématiquement. Dans les deux cas, le point de départ des extensions des grammaires catégorielles classiques est le travail de LAMBEK, initialement développé dans [64].

Le calcul de LAMBEK est un système logique dans lequel les règles universelles des grammaires AB sont vues comme les deux formes (orientées) du *modus ponens*, c'est-à-dire une élimination de l'implication : $A/B\ B \rightarrow A$ correspond à $\Gamma \vdash (B \Leftarrow A),\ \Delta \vdash B \Vdash \Gamma\Delta \vdash A$. Cet aspect logique est accentué par l'ajout des deux règles d'introduction de l'implication, correspondant à $\Gamma A \vdash B \Vdash \Gamma \vdash A \Rightarrow B$. Présenté sous forme de séquents, ce calcul est donc défini de la façon suivante :

$$\frac{}{A \vdash A}\ [ID]$$

$$\frac{\Gamma \vdash A/B \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A}\ [/E] \qquad \frac{\Gamma \vdash B \qquad \Delta \vdash B \backslash A}{\Gamma, \Delta \vdash A}\ [\backslash E]$$

$$\frac{\Gamma, B \vdash A}{\Gamma \vdash A/B}\ [/I] \qquad \frac{B, \Gamma \vdash A}{\Gamma \vdash B \backslash A}\ [\backslash I]$$

Remarque : Il existe plusieurs façons de présenter le calcul de LAMBEK. Nous donnons ici une version simple qui ne tient pas compte du *produit*. Les aspects logique de ce système font l'objet de nombreux travaux, en particulier du fait de ses connexions étroites avec la logique linéaire [49]. Il existe plusieurs variantes de ce calcul, et celui-ci permet une gestion très fine de la sémantique (voir [70], [26]).

La définition d'une grammaire de LAMBEK est en tout point identique à celle d'une grammaire AB. Le langage d'une telle grammaire est défini comme l'ensemble des phrases $w_1 \dots w_n$ telles que

[14] On retrouve cette équivalence en utilisant la forme normale de GREIBACH.

$w_i \triangleright t_i$ pour tout i et $t_1 \dots t_n \vdash s$. Les extensions apportées par les grammaires de LAMBEK offrent une bien meilleure représentation des langues naturelles que les grammaires catégorielles classiques, notamment en permettant de représenter de façon satisfaisante certains phénomènes linguistiques, comme en témoigne l'exemple suivant.

Exemple 1.2. Soit G la grammaire constituée du lexique suivant :
$\{$ *Pierre, Marie, Paul* : SN; *aime, déteste* : $(SN\backslash S)/SN$; *que* : $(SN\backslash SN)/(S/SN)$ $\}$.

La phrase « Paul, que Pierre déteste, aime Marie » appartient au langage de cette grammaire, comme le montre la preuve suivante :

$$\dfrac{\dfrac{SN \vdash SN \quad \dfrac{\dfrac{(SN\backslash S)/SN \vdash (SN\backslash S)/SN \quad \overline{SN \vdash SN}\ ID}{(SN\backslash S)/SN\ SN \vdash SN\backslash S}\ /e}{SN\ (SN\backslash S)/SN\ SN \vdash S}\ \backslash e}{SN\ (SN\backslash S)/SN \vdash S/SN}\ /i \quad \dfrac{(SN\backslash SN)/(S/SN) \vdash (SN\backslash SN)/(S/SN) \quad SN\ (SN\backslash S)/SN \vdash SN\backslash SN}{SN\ (SN\backslash SN)/(S/SN)\ SN\ (SN\backslash S)/SN \vdash SN}\ \backslash e}{SN\ (SN\backslash SN)/(S/SN)\ SN\ (SN\backslash S)/SN\ (SN\backslash S)/SN\ SN \vdash S}\ /e \quad \dfrac{(SN\backslash S)/SN \vdash (SN\backslash S)/SN \quad SN \vdash SN}{(SN\backslash S)/SN\ SN \vdash SN\backslash S}\ /e \quad \backslash e$$

PENTUS a démontré que la classe des langages (de chaînes) des grammaires de LAMBEK est exactement celle des langages algébriques [85], et par conséquent est équivalente aussi à celles des grammaires AB. Ce résultat ne contredit pas le fait que la représentation des langues naturelles soit linguistiquement meilleure avec les grammaires de LAMBEK qu'avec les grammaires AB, car les structures représentant les preuves dans le calcul de LAMBEK peuvent être plus complexes, et donc mieux rendre compte des phénomènes linguistiques décrits. En revanche les grammaires de LAMBEK présentent un inconvénient majeur, démontré également par PENTUS : l'analyse syntaxique (autrement dit la question de savoir si $x \in L(G)$) est un problème NP-complet [86]. Ceci pose bien entendu un problème sérieux quant à d'éventuelles applications pratiques de ce système sur de gros volumes de données.

Afin de mieux répondre aux besoins de la modélisation des langues naturelles, le calcul logique de LAMBEK a évolué ensuite notamment vers les logiques multimodales. Ces systèmes, introduits indépendamment dans [69], [79] et [53], sont capables de rendre compte de façon très précise de phénomènes linguistiques complexes. MOORTGAT donne dans [70] une synthèse assez complète de ce type de systèmes. Cependant nous n'étudierons pas de manière approfondie cette partie pourtant très riche du monde des grammaires catégorielles parce que la base logique de ces formalismes grammaticaux en fait des systèmes assez éloignés des systèmes à base de règles réécriture "simples", pour lesquels l'apprenabilité est moins difficile à obtenir.

Un autre aspect extrêmement important des grammaires catégorielles "modernes" réside dans une gestion très fine de la sémantique. En fait les grammaires catégorielles permettent de dériver en parallèle la sémantique et la syntaxe, précisément à l'aide d'une interprétation en termes de logique des connecteurs (*Principe de compositionnalité*). Basé sur les travaux de MONTAGUE [67], cet aspect des grammaires catégorielles leur donne une légitimité linguistique pour la représentantion et l'étude des langues naturelles. Malgré son intérêt évident, nous n'aborderons pas ce sujet par la suite car les travaux

que nous présentons sont concentrés sur l'acquisition purement syntaxique[15].

1.3.2.3 Grammaires Catégorielles Combinatoires (STEEDMAN)

Les *grammaires catégorielles combinatoires* (*Combinatory Categorial Grammars, CCG*), proposées par STEEDMAN ([102], [103], [104]), sont une autre voie de développement des grammaires catégorielles classiques. Le principe de ce modèle est de définir des règles universelles en adéquation avec la réalité linguistique, mais sans passer nécessairement aux modèles logiques. Les règles ainsi ajoutées sont donc sous la forme de règles de réécriture, telles celles des grammaires AB. On retrouve en partie dans ces règles certaines propriétés du calcul de LAMBEK, du fait de leur pertinence linguistique. De même, la compositionnalité syntaxe/sémantique est un élément central de ce modèle. Voici les règles principales des grammaires catégorielles combinatoires :

Forward Application	$X/Y \quad Y \quad \rightarrow \quad X$
Backward Application	$Y \quad Y\backslash X \quad \rightarrow \quad X$
Forward Composition	$X/Y \quad Y/Z \quad \rightarrow \quad X/Z$
Backward Composition	$Z\backslash Y \quad Y\backslash X \quad \rightarrow \quad Z\backslash X$
Forward Type Raising	$X \quad \rightarrow Y/(X\backslash Y)$
Backward Type Raising	$X \quad \rightarrow (Y/X)\backslash Y$

Remarque : La notation $A\backslash B$ n'a pas la même signification dans le cadre des grammaires catégorielles combinatoires et dans celui des grammaires catégorielles classiques : dans le premier cas la règle d'application arrière est traditionnellement écrite $B \; A\backslash B \quad \rightarrow \quad A$, autrement dit le type "à consommer" (l'argument) dans $A\backslash B$ est B et non A. Ceci dans le but que pour les deux opérateurs $/$ et $\backslash$ le premier type soit toujours le type principal et le second l'argument, tandis que la notation des grammaires catégorielles classiques est basée sur la similarité avec les fractions (l'argument y est "sous" l'opérateur $/$ ou $\backslash$). Afin d'éviter les confusions nous utiliserons toujours la notation des grammaires catégorielles classiques, y compris dans cette partie. Ce choix a bien entendu pour inconvénient d'inverser la notation habituelle des grammaires catégorielles combinatoires.

Les grammaires catégorielles combinatoires sont plus orientées vers les applications informatiques que les grammaires multimodales, dont la complexité algorithmique est importante. En particulier de nombreux travaux ont été réalisés dans ce modèle sur la construction ou l'acquisition de corpus et l'analyse robuste (notamment [55]). Dans la problématique de l'apprenabilité, ce modèle présente l'avantage d'être plus proche des grammaires AB au niveau de son fonctionnement (règles de réécriture et structures de dérivation semblables). Or on verra que les grammaires AB ont certaines propriétés positives pour l'apprentissage, et qu'elles sont donc plus susceptibles d'être adaptables aux grammaires catégorielles combinatoires.

[15] Il sera néanmoins question dans le chapitre 3 de certains travaux sur l'acquisition de grammaires catégorielles à partir d'informations d'ordre sémantique (notamment [43]).

1.3.3 Grammaires de dépendances

> *La phrase est un* ensemble organisé *dont les éléments constituants sont les* mots*. Tout mot qui fait partie d'une phrase cesse par lui-même d'être isolé comme dans le dictionnaire. Entre lui et ses voisins, l'esprit aperçoit des* connexions*, dont l'ensemble forme la charpente de la phrase. [...] Les connexions structurales établissent entre les mots des rapports de* ***dépendance****. (*TESNIÈRE *[107])*

D'un point de vue linguistique, la description de la structure d'une phrase en termes de dépendances est pertinente. En effet, *l'un des aspects les plus importants et les plus visibles du discours humain est le degré d'organisation très élevé des énoncés. [...] Les mots d'un énoncé sont reliés entre eux par des* ***dépendances*** *: un mot peut ou doit dépendre d'un autre pour sa position dans la phrase et sa forme grammaticale. ([66], p.2-3).*

MEL'ČUK donne la définition suivante de la dépendance linguistique [66] :

Définition 1.21. La dépendance est une relation non symétrique, de même type que l'implication : l'un des éléments "présuppose" en quelque sorte l'autre, mais l'inverse n'est (en général) pas vrai. La dépendance est notée par une flèche : $w_1 \rightarrow w_2$ signifie que w_2 dépend de w_1. w_1 est appelé le **gouverneur** de w_2 ; on parle aussi de la **tête** de la dépendance.

1.3.3.1 Les différents types de dépendances

MEL'ČUK distingue quatre niveaux de représentation d'une phrase : *sémantique*, *syntaxique*, *morphologique* et *phonologique*. Pour exprimer formellement la structure d'une phrase sur les trois premiers niveaux de représentation, il propose des graphes étiquetés de la manière suivante : les noeuds représentent des unités linguistiques du niveau correspondant, et les arcs représentent les relations (de dépendance) entre ces unités. On distingue donc trois catégories majeures de dépendances linguistiques :

Dépendances sémantiques : La signification d'une phrase peut être représentée à l'aide du formalisme du calcul des prédicats. On dit alors que l'argument d'un prédicat *dépend sémantiquement* de son prédicat.

Exemple 1.3. Si on considère le prédicat à trois arguments *envoyer(Envoyeur, Objet, Receveur)*, la phrase *"Pierre a envoyé une lettre à Paul"* est décrite au niveau sémantique par la figure 1.2.

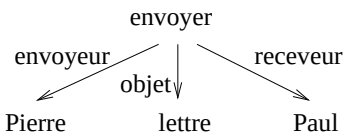


Figure 1.2 – Dépendance sémantique

Dépendances morphologiques : Un mot w_2 *dépend morphologiquement* d'un autre mot w_1 dans une phrase donnée si au moins une des "catégories grammaticales" de w_2 est déterminée par w_1. Un exemple simple de dépendance morphologique est l'accord singulier/pluriel entre un nom et son déterminant. Ce type de dépendance est très variable selon la langue.

Dépendances syntaxiques : La *structure syntaxique* d'une phrase est d'une certaine manière un intermédiaire entre sa structure sémantique (signification, ou *contenu* de la phrase) et sa structure morphologique (*forme* de la phrase). Cette structure intermédiaire permet de décrire *l'organisation* de la phrase. MEL'ČUK explique que ce statut *intermédiaire* rend la notion de structure syntaxique plus difficile à définir, et met en garde contre le risque de confusion avec les deux autres types de structures à cause de cette difficulté. Mais cet aspect intermédiaire rend aussi la structure syntaxique plus intéressante à étudier, puisqu'elle est plus riche en "information" que la structure morphologique, sans avoir la trop grande complexité de la structure sémantique. Beaucoup de travaux sur les grammaires de dépendances utilisent ce niveau syntaxique (parfois implicitement), et c'est aussi sur celui-ci que nous travaillons.

1.3.3.2 Les dépendances syntaxiques [66]

Les dépendances syntaxiques ont été utilisées pour parler de la structure des phrases depuis l'antiquité. On retrouve cette notion aussi bien chez les grammairiens arabes du $8^{\text{ème}}$ siècle que dans des livres de grammaire allemands du $19^{\text{ème}}$. Mais la représentation en constituants (ou *"phrase structure"*) est apparue au début du $20^{\text{ème}}$, alors que l'approche avec dépendances syntaxiques n'a été formellement présentée qu'en 1959 par TESNIÈRE [107]. Comme la classification de Chomsky est basée sur cette approche en constituants, l'approche avec dépendances n'est devenue populaire qu'au début des années 1980.

MEL'ČUK précise que, malgré son caractère intermédiaire entre les structures sémantique et morphologique et la difficulté à la définir, la structure syntaxique ne doit pas être vue comme une abstraction linguistique, sans rapport avec la réalité psychologique. Il pense au contraire que cette structure est un *"modèle raisonnable de ce qui se produit dans le cerveau du locuteur"*.

MEL'ČUK propose des critères linguistiques permettant de déterminer s'il y a une dépendance syntaxique entre deux mots w_1 et w_2 d'une phrase, et si oui quel est son sens (lequel des deux mots est le gouverneur de la dépendance) et quel est le type de cette dépendance. Nous ne détaillons pas ici ces critères ; l'exemple ci-dessous illustre sur un cas particulier le type de raisonnement que propose MEL'ČUK sur les dépendances syntaxiques.

Exemple 1.4. Il est d'usage dans les grammaires catégorielles de considérer que dans une expression *Déterminant + Nom*, le gouverneur doit être le déterminant (ce qui se traduit par un type SN/N pour le déterminant). MEL'ČUK affirme au contraire que la dépendance entre un nom et son déterminant doit être gouvernée par le nom, principalement pour les raisons suivantes :

- La *valence syntaxique passive* (c'est-à-dire l'ensemble des fonctions grammaticales que peut prendre l'expression) d'une expression comme *the dog* est celle du nom *dog* et non celle de l'article *the*. Par exemple, si l'expression *the dog* est employée comme sujet, on voit bien que le rôle de sujet est plus celui de *dog* que celui de *the*. En effet, le verbe s'accorde avec le nom et pas avec le déterminant : *the dogs are barking* est correct en anglais, alors que *the dogs is barking* ne l'est pas.

- Il est nécessaire de refléter le parallélisme dans le comportement syntaxique d'expressions telles que ***the*** *dog*, ***this*** *dog* et ***Alan's*** *dog* : si on considère que le déterminant gouverne l'expression, alors on doit aussi accepter dans ces deux derniers cas que *this* et *Alan's* soient la tête de *dog*.

1.3.3.3 Grammaires catégorielles de dépendances

Si l'on considère une interprétation étendue de la définition des grammaires de dépendances, les grammaires catégorielles peuvent être vues comme une forme de grammaires de dépendances. En effet, les opérateurs $/$ et $\backslash$ sont orientés : lors d'une étape de dérivation $A/B\ B\ \rightarrow A$ (resp. $B\ B\backslash A\ \rightarrow A$), le constituant de type B dépend du constituant de type A/B (resp. $B\backslash A$). On peut aussi ramener les dépendances au niveau des mots en descendant par induction dans le sous-arbre de chaque constituant : chaque constituant a une unique tête, donc lorsqu'un constituant dépend d'un autre il existe une dépendance depuis la tête du second jusqu'à la tête du premier[16].

DIKOVSKY propose un modèle de grammaires de dépendances basé sur le même fonctionnement que les grammaires catégorielles. Ce système nommé *grammaires catégorielles de dépendances* est présenté dans [40] et [41]. Sa particularité réside dans la gestion de dépendances distantes à l'aide de catégories "libres", mais potentiellement contraintes par un *ancrage* particulier.

$\mathbf{C}$ représente un ensemble fini de *catégories élémentaires*. Chaque catégorie élémentaire peut être *itérée* et devenir *optionnelle*. $\mathbf{C}^* =_{df} \{X^* \mid X \in \mathbf{C}\},\ \mathbf{C}^+ =_{df} \{X^+ \mid X \in \mathbf{C}\},\ \mathbf{C}^? =_{df} \{X^? \mid X \in \mathbf{C}\}$, représentent respectivement les ensembles de catégories *itératives*, *répétitives* et *optionnelles*. $\mathbf{C}^\omega =_{df} \mathbf{C}^* \cup \mathbf{C}^+ \cup \mathbf{C}^?$. Toutes les catégories contenues dans $\mathbf{C}^\omega$ sont *neutres*. Par ailleurs des catégories *polarisées* sont définies, avec l'une des quatre polarités orientées suivantes : gauche et droite positive $\nwarrow, \nearrow$ et gauche et droite négative $\searrow, \swarrow$. Pour chaque polarité v, il y a une unique polarité "duale" $\breve{v}$: $\breve{\nwarrow} = \swarrow$, $\breve{\swarrow} = \nwarrow$, $\breve{\nearrow} = \searrow$, $\breve{\searrow} = \nearrow$. Intuitivement, les catégories positives peuvent être vues comme des valences des dépendances distantes sortant du gouverneur, et les catégories négatives comme celles des dépendances distantes entrant dans les mots subordonnés. Elles correspondent donc respectivement aux débuts et aux fins des dépendances distantes. Par exemple, la valence positive $\nwarrow wh\_upon\_obj$ marque le début de la dépendance distante wh_upon_obj d'un verbe transitif qui gouverne le syntagme WH déplacé à gauche gouverné par la préposition 'UPON', tandis que la fin de cette dépendance est marquée par la valence négative duale $\swarrow wh\_upon\_obj$ de cette préposition (cf. *upon what dependency theory we rely*).

$\nearrow \mathbf{C}, \nwarrow \mathbf{C}, \searrow \mathbf{C}$ et $\swarrow \mathbf{C}$ représentent les ensembles correspondant aux catégories de dépendances distantes polarisées. Par exemple, $\nearrow \mathbf{C} = \{(\nearrow C) \mid C \in \mathbf{C}\}$ est l'ensemble des catégories *positives à droite*. $V^+(\mathbf{C}) = \nearrow \mathbf{C} \cup \nwarrow \mathbf{C}$ est l'ensemble des catégories de dépendances distantes positives, $V^-(\mathbf{C}) = \searrow \mathbf{C} \cup \swarrow \mathbf{C}$ est l'ensemble des négatives.

Dans l'utilisation des dépendances distantes, il est parfois nécessaire d'exprimer certaines contraintes sur la position du mot subordonné : s'il doit être le premier (dernier) mot de la phrase, du constituant, etc., ou s'il doit immédiatement précéder (suivre) un certain mot. Par exemple, en français, la catégorie négative $\swarrow clit-dobj$ d'un clitique objet direct doit être ancrée au verbe auxiliaire ou à un verbe en

[16] Le lien entre grammaires de dépendances et grammaires catégorielles est étudié plus en détail dans [74].

forme non-analytique. Dans ce but nous distinguons dans l'ensemble des catégories de dépendances distantes négatives le sous-ensemble $Anc(\mathbf{C}) \subseteq V^-(\mathbf{C})$ des catégories négatives *ancrées* des autres catégories appelées *libres*.

Définition 1.22. L'ensemble $Cat(\mathbf{C})$ des catégories de structures de dépendances (SD) est le plus petit ensemble vérifiant les conditions :
1. $\mathbf{C} \cup V^-(\mathbf{C}) \subset Cat(\mathbf{C})$.
2. Pour $C \in Cat(\mathbf{C})$, $A_1 \in \mathbf{C}^\omega \cup \nwarrow \mathbf{C}$, $A_2 \in \mathbf{C}^\omega \cup \nearrow \mathbf{C}$, et $B \in Anc(\mathbf{C})$, les catégories $[A_1 \backslash C]$, $[C / A_2]$, $[B \backslash\backslash C]$ et $[C /\!/ B]$ appartiennent aussi à $Cat(\mathbf{C})$.

Nous supposons que tous les constructeurs $\backslash$, $/$, $\backslash\backslash$, $/\!/$ sont associatifs. Ainsi toute catégorie d'une SD α peut être présentée sous la forme

$$\alpha = [L_k\ l_k\ \ldots\ L_1\ l_1\ C\ r_1\ R_1\ \ldots\ r_m\ R_m],$$

où l_i et r_j sont respectivement des constructeurs gauches et droits. Par exemple,

$$[(\swarrow clit - dobj) \backslash\backslash subj \backslash S / auxPP]$$

est l'une des catégories d'un verbe auxiliaire, qui le définit comme le mot d'appui pour un clitique objet direct, nécessite une dépendance locale à gauche pour le sujet et une dépendance locale à droite $auxPP$ pour un participe passé subordonné.

Définition 1.23. Une grammaire catégorielle de dépendances (GCD) est un système $G = (W, \mathbf{C}, S, \delta)$, où W est un ensemble fini de mots, $\mathbf{C}$ un ensemble fini de catégories élémentaires contenant la catégorie racine S, et δ - le lexique - est une substitution finie sur W telle que $\delta(a) \subset Cat(\mathbf{C})$ pour tout mot $a \in W$.

Nous étendons maintenant à de nouveaux types les définitions de langage de chaînes et de structures générés par une GCD. La partie centrale de cette définition porte sur les règles qui régissent l'utilisation des valences de dépendances polarisées. L'idée qui sous-tend ces règles est qu'il faut, pour établir une dépendance distante entre deux mots de valences duales, que ces deux valences soit *chargées*. Les valences positives et libres sont chargées par définition. Les valences ancrées sont initiallement déchargées. Dès que la position correcte du mot ancré subordonné est identifiée, sa valence devient chargée et donc disponible pour le gouverneur de la dépendance. Dans le but de distinguer les valences chargées des valences déchargées, nous utilisons pour chaque valence de dépendance vC une unique copie chargée $\#(vC)$.

Le langage de chaînes et le langage de structures générés par une GCD sont définis à l'aide d'une relation de prouvabilité $\vdash$ sur les séquences de catégories. Nous supposerons que toute occurrence d'une catégorie $\Gamma_1 C \Gamma_2$ correspond à une structure de dépendances D de catégorie C. $r(D)$ représente la racine de D. Dans la suite $\vdash$ n'est définie que pour les constructeurs gauches. Les règles pour les constructeurs droits sont symétriques.

Définition 1.24. Relation de prouvabilité $\vdash$: [17]
Règle de dépendances locales :

[17] $\vdash$ définit aussi une relation sur les couples (D, Γ) où D est explicité à chaque règle.

L. $\Gamma_1 C[C\backslash\alpha]\Gamma_2 \vdash \Gamma_1\alpha\Gamma_2$. Si C est la catégorie de D_1 et $[C\backslash\alpha]$ est celle de D_2, alors α devient la catégorie de la nouvelle SD : $D_1 \cup D_2 \cup \{r(D_1) \overset{C}{\longleftarrow} r(D_2)\}$.

Règles d'ω-dépendances :

I. $\Gamma_1 C[C^*\backslash\alpha]\Gamma_2 \vdash \Gamma_1[C^*\backslash\alpha]\Gamma_2$. Si C est la catégorie de D_1 et $[C^*\backslash\alpha]$ est celle de D_2, alors $[C^*\backslash\alpha]$ dans la conséquence devient la catégorie de la nouvelle SD : $D_1 \cup D_2 \cup \{r(D_1) \overset{C}{\longleftarrow} r(D_2)\}$.

R. $\Gamma_1 C[C^+\backslash\alpha]\Gamma_2 \vdash \Gamma_1[C^*\backslash\alpha]\Gamma_2$. Si C est la catégorie de D_1 et $[C^+\backslash\alpha]$ est celle de D_2, alors $[C^*\backslash\alpha]$ devient la catégorie de la nouvelle SD : $D_1 \cup D_2 \cup \{r(D_1) \overset{C}{\longleftarrow} r(D_2)\}$.

O. $\Gamma_1 C[C^?\backslash\alpha]\Gamma_2 \vdash \Gamma_1\alpha\Gamma_2$. Si C est la catégorie de D_1 et $[C^?\backslash\alpha]$ est celle de D_2, alors α dans la conséquence devient la catégorie de la nouvelle SD : $D_1 \cup D_2 \cup \{r(D_1) \overset{C}{\longleftarrow} r(D_2)\}$.

Ω. $\Gamma_1[C\backslash\alpha]\Gamma_2 \vdash \Gamma_1\alpha\Gamma_2$ pour tout $C \in \mathbf{C}^* \cup \mathbf{C}^?$ [18].

Règle des dépendances ancrées :

A. $\Gamma_1 C[C\backslash\!\backslash\alpha]\Gamma_2 \vdash \Gamma_1\#(C)\alpha\Gamma_2$ pour $C \in Anc(\mathbf{C})$.

Règle des dépendances distantes :

D. $\Gamma_1\#(\swarrow C)\Gamma_2[(\nwarrow C)\backslash\alpha]\Gamma_3 \vdash \Gamma_1\Gamma_2\alpha\Gamma_3$. La règle s'applique s'il n'y a aucune occurrence des catégories $\swarrow C$ et $\nwarrow C$ dans Γ_2. Si $\swarrow C$ est la catégorie de D_1 et $[(\nwarrow C)\backslash\alpha]$ est celle de D_2, alors α devient la catégorie de la nouvelle SD : $D_1 \cup D_2 \cup \{r(D_1) \overset{C}{\longleftarrow} r(D_2)\}$.

$\vdash^*$ est définie comme la fermeture réflexive et transitive de $\vdash$.

Définition 1.25. Une SD D est associé par une GCD $G = (W, \mathbf{C}, S, \delta)$ à une phrase w (noté $G(D, w)$) si D est défini comme une SD de catégorie S dans une preuve $(D_\emptyset, \Gamma) \vdash^* (D, S)$ où $D_\emptyset$ désigne la SD initiale sans arête et $\Gamma \in \delta(w)$.
Le langage de structures généré par G est l'ensemble des SD : $\Delta(G) = \{D \mid \exists w \in W^+ \; G(D, w)\}$.
Le langage de chaînes généré par G est l'ensemble des phrases : $\mathcal{L}(G) = \{w \in W^+ \mid \exists D \; G(D, w)\}$.

1.3.3.4 Grammaires de liens

Les grammaires de liens (*link grammars*) sont définies par Sleator et Temperley dans [98]. Il s'agit d'un formalisme relativement simple, qui permet néanmoins de représenter de manière précise un langage naturel. Les auteurs ont ainsi modélisé la langue anglaise dans ce formalisme, de manière à gérer la plupart des aspects grammaticaux de cette langue. Ils ont aussi réalisé un analyseur syntaxique basé sur ce modèle [106]. Dans notre perspective de l'apprentissage basé sur les grammaires de type grammaires catégorielles, le travail de Sleator et Temperley est particulièrement intéressant pour les raisons suivantes, aussi bien théoriques que pratiques :

- de manière générale, la simplicité du modèle des grammaires de liens offre des possibilités très intéressantes d'adaptation à différents objectifs, tout en ayant une capacité de représentation du langage assez robuste ;
- il existe un rapport assez étroit entre les grammaires de liens et les grammaires catégorielles classiques : ces deux formalismes sont totalement lexicalisés, et fondés sur une notion de dépendance ;

[18] Les SD restent inchangées lorsque le contraire n'est pas spécifié.

- il est possible d'exprimer le modèle des grammaires de liens à l'aide d'un petit ensemble de règles basées sur l'unification. Ce point est essentiel pour notre utilisation de ce modèle dans le cadre de l'apprentissage partiel.

Le principe général des grammaires de liens est qu'une phrase est correcte s'il est possible d'établir des liens entre les mots, selon les contraintes imposées par les définitions des mots dans le lexique. Ces liens représentent les relations syntaxiques entre les mots. Une grammaire de liens est définie par un vocabulaire, un ensemble de connecteurs et une relation qui met en correspondance chaque mot avec un ensemble de types. Un type est un couple de listes ordonnées de connecteurs. Une phrase donnée appartient au langage décrit par la grammaire de liens s'il est possible de dessiner tous ses liens au dessus des mots de manière à satisfaire les critères suivants :

Planarité : Les liens (dessinés au dessus des mots) ne se croisent pas[19].

Connectivité : Il existe un chemin (de liens) entre tout couple de mots de la phrase.

Ordre : Pour chaque mot w de la phrase, il existe un type qui lui est associé dans le lexique tel que ce type est saturé. Un type $d = \texttt{d([}L_1\texttt{,..,}L_l\texttt{], [}R_1\texttt{,..,}R_r\texttt{])}$ est saturé si pour tout connecteur L_i (resp. R_i) il existe un lien étiqueté par le nom du connecteur venant de gauche (resp. de droite) vers w, et pour tout couple de connecteurs L_i, L_j (resp. R_i, R_j) si $i < j$ alors le mot connecté par L_i (resp. R_i) est plus proche de w que celui connecté par L_j (resp. R_j).[20]

Exclusion : Deux liens ne peuvent pas relier le même couple de mots.

Ces critères sont appelés les méta-règles des grammaires de liens. Les auteurs montrent dans [98] que la classe de langages des grammaires de liens est équivalente à celle des grammaires hors-contexte.

Exemple 1.5. Avec le lexique suivant, la phrase ci-dessous est correcte :

a,the ▷ `d([],[D])`
cat, snake ▷ `d([D],[S])`, `d([O,D],[])`
chased ▷ `d([S],[O])`

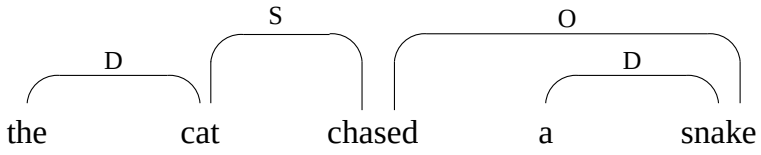


Figure 1.3 – Dérivation avec les grammaires de liens

[19] Formellement : s'il existe deux liens l et l' entre deux couples de mots (w_i, w_j) et $(w_{i'}, w_{j'})$ (où w_k est le k^{eme} mot de la phrase), alors l'une des inégalités suivantes est vérifiée : $i < j \leq i' < j'$ ou $i' < j' \leq i < j$.

[20] *Remarque :* dans la définition originale la liste de connecteurs à droite est notée $\texttt{[}R_r\texttt{,...,}R_1\texttt{]}$. Nous choisissons ici d'inverser l'ordre pour maintenir la cohérence avec la notation `cons(c,L)/nil` présentée dans la partie 4.3.4.

1.4 Conclusion

Dans ce chapitre nous n'avons fait qu'esquisser quelques-uns des principaux formalismes grammaticaux adaptés aux langues naturelles. Nous en avons omis de nombreux autres, parmi lesquels on peut citer notamment les grammaires d'arbres adjoints (TAG) [58], les grammaires minimalistes [101], les LFG, HPSG [24], [89], etc. Ces omissions volontaires sont liées à la nature de notre objectif dans cette étude : nous ne recherchons pas l'exhaustivité ni même la meilleure façon de représenter et/ou traiter des langues naturelles en général. Notre objectif étant l'utilisation de formalismes grammaticaux (adaptés aux langues naturelles) dans le cadre de l'apprentissage automatique, l'éventail des modèles traités se réduit à ceux présentant des prédispositions pour cette tâche.

Or nous verrons dans le chapitre 2 que le point de vue de l'apprenabilité est très restrictif sur les classes de langages. Celles-ci doivent en effet respecter des propriétés contraignantes, ce qui rend assez difficile la découverte de classes de langages apprenables qui soient à la fois exprimées dans un formalisme adéquat pour les langues naturelles, et suffisamment expressives par rapport à celles-ci. Les grammaires catégorielles classiques ont été le premier système grammatical étudié avec succès dans cette perspective. Nous reviendrons plus en détail dans le chapitre 3 sur les résultats obtenus par KANAZAWA [60] à ce niveau. Néanmoins les grammaires catégorielles classiques sont relativement pauvres pour représenter les langues naturelles, d'où l'intérêt de chercher à étendre ces bons résultats à des formalismes plus riches.

C'est la raison pour laquelle les formalismes grammaticaux que nous présentons restent relativement proches des grammaires catégorielles (au sens large) : nous espérons avec ceux-ci pouvoir tirer profit des résultats obtenus par KANAZAWA, en nous inspirant notamment de sa méthode pour les grammaires AB. Plus précisément, les formalismes dont nous traitons possèdent certains points communs susceptibles d'être favorables à leur apprenabilité :

- la lexicalisation des grammaires semble un avantage important pour l'apprentissage. Nous verrons toutefois qu'il ne s'agit nullement d'une condition nécessaire.
- la possibilité de contrôler les dérivations à l'aide de règles de réécriture, éventuellement avec des substitutions de variables : c'est typiquement le cas des grammaires AB définies par deux règles universelles, applicables par substitution de types aux variables de la règle. Le rôle de l'unification est d'une importance particulière dans la méthode d'apprentissage développée par KANAZAWA.

Afin de factoriser autant que possible ces ressemblances et d'atteindre un niveau de généralisation élevé, nous proposerons dans le chapitre 4 un modèle de représentation des formalismes grammaticaux, basé sur celui introduit par KANAZAWA dans [60]. Ce modèle reprend le contrôle des dérivations par des règles de réécriture universelles, tout en étendant la définition des grammaires de façon à autoriser une souplesse maximale. Comme les résultats d'apprenabilité que nous présenterons ensuite seront exprimés à travers ce modèle, la possibilité d'exprimer les formalismes définis dans le présent chapitre sous cette forme rejoint les arguments déjà évoqués en leur faveur.

CHAPITRE 2

Inférence grammaticale : le modèle de Gold

LE MODÈLE D'IDENTIFICATION À LA LIMITE, AUSSI APPELÉ DU NOM DE SON AUTEUR MODÈLE DE GOLD, EST L'UNE DES PRINCIPALES REPRÉSENTATIONS FORMELLES DU PROCESSUS D'APPRENTISSAGE. LA PREMIÈRE DÉFINITION EN EST DONNÉE PAR GOLD DANS [50]. L'AUTEUR LUI-MÊME EST D'ABORD PESSIMISTE QUANT À L'INTÉRÊT DE CE MODÈLE, À CAUSE DE L'APPARENTE IMPOSSIBILITÉ D'Y OBTENIR DES RÉSULTATS POSITIFS POUR DES CLASSES DE LANGAGES "INTÉRESSANTES". PLUS TARD, LES RÉSULTATS POSITIFS D'ANGLUIN AVEC CE MODÈLE MONTRERONT SA PERTINENCE. LE MODÈLE SERA ENSUITE ÉTUDIÉ PLUS EN DÉTAIL : AINSI, PLUSIEURS AUTRES RÉSULTATS ENCOURAGEANTS VIENDRONT SOUTENIR L'IDÉE QUE L'IDENTIFICATION À LA LIMITE CONSTITUE BIEN UN CADRE THÉORIQUE ADAPTÉ À LA REPRÉSENTATION DU PROCESSUS D'APPRENTISSAGE, EN PARTICULIER CELUI DES LANGAGES, VOIRE DES LANGUES NATURELLES.

2.1 Introduction

L'*inférence grammaticale* désigne la problématique qui consiste à apprendre des langages à partir de données. Mais qu'entend-on par *apprentissage*, *langages* et *données* ? Tout cadre formel pour ce problème doit donc avant tout répondre aux questions essentielles suivantes, qui justement différencieront les modèles d'inférence grammaticale (et leurs éventuelles applications) :

- Quelle est la nature des données dont on dispose ? Le cas le plus simple est sans doute celui de chaînes composées à partir d'un alphabet fini, mais il peut aussi s'agir d'arbres, de termes, de graphes ou de tout autre type de structures. On peut aussi à ce niveau s'interroger sur la quantité, la qualité, la complétude des données. Cette question est de plus étroitement liée à la question du type de langages considérés, ainsi qu'à la faisabilité du problème en termes d'applications.
- Quel est le type de langages considéré, et comment les langages sont-ils représentés ? Quelles restrictions impose-t-on aux langages pour pouvoir répondre au problème, sans pour autant le vider de tout intérêt ? On ne peut bien entendu pas dissocier l'apprentissage de langages de celui de grammaires, ces dernières étant indispensables pour décrire les langages. Ici *grammaire* doit

être pris au sens le plus large, à savoir moyen de représentation d'un langage. On peut étudier l'apprentissage de langages dans un contexte très général sans préciser aucune contrainte (ou très peu) sur la relation qui unit grammaires et langages, ou au contraire s'intéresser à une forme spécifique, comme par exemple des automates ou l'un des nombreux formalismes grammaticaux adaptés au langage naturel.

- Quelle est la nature du processus d'inférence, et qu'en attend-on ? Le processus peut être fini ou non. La solution peut être unique, ou au contraire multiple. Le processus peut être totalement automatique ou semi-automatique, auquel cas il est nécessaire qu'un humain le guide. Ceci peut alors prendre différentes formes : correction d'erreurs, réponse à des questions, etc. Il faut aussi définir un critère de succès du processus : obtention d'une grammaire très précise ou d'une approximation, limites éventuelles sur le temps, nombre d'essais autorisés ou complexité algorithmique du processus.

L'inférence grammaticale peut donc prendre des formes diverses, selon les caractéristiques du problème étudié. En tant que domaine de recherche, l'inférence grammaticale est ainsi liée à de nombreux autres domaines : apprentissage automatique, théorie des langages formels, traitement automatique du langage naturel, reconnaissance de motifs et bioinformatique. Une part importante des travaux porte uniquement sur les langages réguliers, ou des sous-classes de ceux-ci (voir [44]). DE LA HIGUERA [39] distingue les cinq sous-domaines suivants, qui font l'objet de recherches en inférence grammaticale :

- *L'inférence inductive* est définie de manière très générale comme l'identification d'une fonction à partir des valeurs de celle-ci. ANGLUIN et SMITH soulignent la différence entre *inférence inductive* et *apprentissage* :

  > *"Il est important de ne pas confondre inférence inductive et apprentissage, même si l'induction peut être utilisée en apprentissage. Selon le dictionnaire Webster "apprendre" consiste à acquérir de la connaissance, de la compréhension, de l'instruction ou de l'expérience", ou plus généralement "devenir capable de". En comparaison, "l'induction" est définie comme "l'acte, le processus ou le résultat d'une étape de raisonnement de la partie vers le tout, du particulier au général, ou de l'individuel à l'universel". [8]*

  L'inférence grammaticale par induction peut être vue elle-même comme une instance de l'inférence inductive, dans laquelle les phrases du langage sont les *parties* (valeurs de la fonction) qui doivent être *généralisées* sous forme d'une grammaire représentant le langage complet (la fonction à trouver). L'inférence inductive peut être considérée comme le cadre le plus large dans lequel l'inférence grammaticale peut avoir lieu. Le modèle de référence dans ce contexte est le modèle défini par GOLD dans [50], dont nous détaillons les caractéristiques dans ce chapitre et sur lequel sont basés tous les résultats que nous proposons.

- *L'inférence polynomiale.* L'efficacité étant une question essentielle de tout problème algorithmique, un grand nombre de travaux portent sur l'existence (ou l'inexistence) d'algorithmes d'apprentissage de complexité polynomiale. Cette question est bien entendu liée au modèle choisi, et amène elle-même à la définition de modèles d'apprentissages spécifiques (par exemple [38]).

Comme l'inférence est un problème généralement très complexe, l'apprenabilité en temps polynomial ne s'applique qu'à des classes de langages très restreintes (sous-classes de langages réguliers/automates déterministes).

- *L'apprentissage actif* concerne l'ajout dans le processus d'apprentissage de la possibilité d'effectuer des requêtes à un oracle. Différents types de requêtes sont envisageables (voir [6], [7]). Cette aide à l'apprentissage permet d'élargir le champ des langages apprenables de manière significative. Le nombre de requêtes apparaît dans ce cadre en tant que paramètre supplémentaire du processus, qu'il convient de minimiser.

- *Le modèle PAC*[1] proposé par Valiant dans [108] est utilisé comme modèle d'apprentissage statistique. Dans ce cas, le fait qu'il ne soit pas nécessaire d'obtenir un résultat exact est supposé faciliter l'apprentissage. Cependant peu de résultats positifs ont été obtenus en inférence grammaticale [39].

- *Le modèle PAC simplifié* est (comme son nom l'indique) une simplification du modèle PAC de Valiant, qui a notamment pour avantage de permettre l'apprentissage des automates déterministes ([84]).

La mise en œuvre d'une forme d'apprentissage par la machine suppose en premier lieu que le concept d'apprentissage lui-même soit défini de manière formelle. Cette formalisation est l'objet de l'article de référence de GOLD (publié en 1967), [50]. Celui-ci y propose plusieurs versions du modèle, qui ont toutes en commun de définir comme critère de succès (c'est-à-dire d'apprenabilité) *l'identification à la limite*[2].

Le principe de l'identification à la limite est la convergence : à partir d'une séquence infinie d'éléments qui caractérisent le langage à deviner, l'apprenant émet des hypothèses. Ces hypothèses prennent la forme d'une grammaire, censée correspondre au langage observé jusqu'alors par l'apprenant. Comme l'énumération est infinie, l'apprenant répond lui aussi sous forme d'une séquence infinie de grammaires hypothèses. Finalement, l'apprentissage est réussi si, à partir d'un certain point, l'apprenant émet toujours la même hypothèse (convergence), et que celle-ci correspond bien au langage attendu. Le fait que l'apprenant ignorera toujours s'il a atteint ou non la solution est un aspect important de ce formalisme. GOLD en donne la justification d'ordre linguistique suivante : *"une personne ne sait jamais si elle parle correctement un langage : il y a toujours la possibilité qu'elle découvre que sa grammaire comporte une erreur. Mais on peut garantir qu'un enfant peut apprendre une langue naturelle."* [50].

Dans [50], GOLD propose en fait plusieurs modèles d'apprentissage, ou plus exactement plusieurs versions du modèle d'identification à la limite. Ces versions se distinguent principalement par la *méthode de présentation de l'information* d'une part, et la *relation de dénomination des langages* d'autre part.

[1] *Probably Approximately Correct.*

[2] On peut noter cependant que Gold n'a pas introduit ce concept (il cite notamment lui-même [2]). Selon COSTA FLORÊNCIO [35], le premier travail formel dans cette direction serait celui de Moore [68].

Il y a deux méthodes de présentation : la *présentation par texte*, avec laquelle l'apprenant ne dispose que des *exemples positifs*, c'est-à-dire les éléments du langage uniquement. Avec la seconde méthode de présentation, dite *présentation complète* ou *présentation par informateur*, l'apprenant dispose à la fois des exemples *positifs* et *négatifs*. Cela signifie que chaque exemple donné est accompagné d'une information indiquant si celui-ci appartient au langage décrit ou non. Cette forme d'apprentissage est très peu utilisée pour plusieurs raisons :

- Il est communément admis dans la communauté linguistique que les enfants acquièrent le langage quasiment uniquement à partir d'exemples positifs (voir par exemple [88], [65]). En particulier l'enfant utilise très peu les éventuelles corrections indiquées par son entourage. Ce point constitue un argument pour n'utiliser que la présentation par texte si l'on souhaite modéliser (ou approcher) l'acquisition de la langue par l'enfant.
- GOLD démontre qu'il est possible d'apprendre des classes de langages très larges par présentation complète : si l'ensemble des langages récursifs n'est pas apprenable même dans ce cas, la classe des langages *primitifs récursifs* (qui contient notamment tous les langages contextuels) l'est. Ceci peut être vu comme un argument contre l'apprentissage par présentation complète, au sens où cet apprentissage semble "trop facile". Le fait qu'une classe de langages aussi complexe soit apprenable de manière triviale semble en effet indiquer que le modèle est inadapté.
- Enfin, si l'on souhaite que le modèle d'apprentissage puisse avoir un minimum d'applications dans le monde réel, il est clair que la présentation complète est un obstacle de taille : il est en effet difficilement envisageable en pratique d'obtenir des données négatives (et *a fortiori* les données négatives complètes) pour un langage.

Pour toutes ces raisons, le modèle de GOLD est souvent assimilé à sa version "présentation par texte". De même, tous les résultats que nous proposons par la suite sont aussi obtenus dans ce cadre.

La *relation de dénomination des langages* est le moyen d'associer un "nom" à un langage. Typiquement ces "noms de langages" sont des grammaires, ce que note GOLD dans son article. Deux types de relation sont considérées, le *testeur* et le *générateur*. Un testeur est un algorithme qui répond au problème de l'appartenance d'un objet au langage, tandis qu'un générateur est un algorithme qui énumère l'ensemble des phrases du langage. Un testeur existe si et seulement si le langage est récursif, et un générateur existe si et seulement si le langage est récursivement énumérable. Ainsi cette caractéristique dépend totalement de la classe de langages considérée, et n'a par conséquent aucune incidence sur la définition du modèle lui-même. On verra néanmoins dans ce chapitre (partie 2.3.2) que la restriction aux langages récursifs facilite grandement la caractérisation de langages apprenables.

La principale difficulté de l'apprentissage à partir d'exemples positifs est la *surgénéralisation*. La surgénéralisation est l'erreur qui consiste à trop généraliser (extrapoler) à partir des données fournies, comme l'indique SAKAKIBARA dans [93] : *"Le problème est d'éviter la surgénéralisation, ce qui signifie inférer un langage qui est un sur-ensemble strict du langage inconnu."* Par exemple, on peut supposer que la séquence $3, 5, 7, x$ décrit l'ensemble des nombres impairs supérieurs à 2, et en déduire que x aura pour valeur 9. Mais s'il s'avère que x est 11, la séquence représentant en fait les nombres premiers strictement supérieurs à 2, alors il y a eu surgénéralisation : l'ensemble des nombres représenté est un

sous-ensemble (strict) de l'ensemble qu'on a proposé. Comme on ne dispose que d'exemples positifs, il n'y aura jamais de contre-exemple dans la séquence permettant de corriger l'erreur. Bien entendu, la généralisation est indispensable dans le processus d'apprentissage. En effet, une méthode d'apprentissage "trop prudente" qui ne généraliserait jamais ne ferait pas véritablement un *apprentissage,* au sens d'une découverte de quelque chose de nouveau : il s'agirait simplement d'une sorte de compilation des exemples proposés. Surtout, il est évident qu'une telle méthode serait incapable d'identifier un langage infini.

Sauf cas particuliers, la généralisation doit donc bien être utilisée au cours de l'apprentissage. La question qui se pose d'un point de vue algorithmique est : quand faut-il généraliser ? (ou quand faut-il ne pas généraliser, selon ce qu'on considère comme étant l'action par défaut). Mais avant de se poser cette question, il faut s'assurer qu'il est *possible de savoir quand généraliser*, car lorsqu'on ne dispose pas d'exemples négatifs on n'a aucun indice sur la position de la frontière du langage à deviner. C'est précisément ce point qui pose problème au départ avec l'identification à la limite à partir d'exemples positifs : GOLD constate immédiatement après avoir défini son modèle que même les classes de langages qu'on croyait simples (les langages réguliers, voir exemple 2.2) ne sont pas apprenables, parce qu'on ne peut pas savoir quand [ne pas] généraliser.

La caractérisation de classes de langages apprenables passera donc d'abord par l'étude des cas où cette question peut être résolue. C'est seulement au début des années 1980 qu'ANGLUIN apporte au modèle de GOLD ses premiers résultats positifs : dans [5], elle propose de *"considérer le cas particulier d'inférence à partir d'exemples positifs qui évite la surgénéralisation [et donne] des conditions suffisantes pour cela."* Le critère proposé par ANGLUIN, basé sur les *ensembles révélateurs*, a donné lieu à de nombreuses utilisations ou extensions démontrant finalement la richesse du modèle de GOLD. L'une des propriétés dérivées de ce premier résultat positif est *l'élasticité finie* proposée par WRIGHT, elle-même réutilisée avec succès par SHINOHARA dans la propriété de *densité finie bornée*.

Ce chapitre propose un panorama du modèle de GOLD, à travers les définitions et principaux résultats "d'apprenabilité générale" dans ce modèle, c'est-à-dire les résultats qui ne relèvent pas d'un formalisme particulier. Tous les théorèmes et propositions sont accompagnés de leur démonstration de manière à fournir au lecteur une vision complète de chaque résultat, notamment du point de vue algorithmique. On utilisera en effet beaucoup certaines propriétés, comme l'élasticité finie, pour démontrer qu'une classe de langages est apprenable : les démonstration de la partie 2.4.1 décrivent un algorithme "universel" d'apprentissage pour les classes possédant cette propriété.

2.2 Définition et conséquences immédiates

2.2.1 Terminologie

2.2.1.1 Langages et objets

Dans la littérature du domaine, la définition du *langage* est fréquemment réduite à un ensemble de phrases, chaque phrase étant une simple séquence finie de mots du vocabulaire. C'est notamment le

cas dans [50], [5] et [61] : un langage y est défini comme un sous-ensemble de l'ensemble de toutes les séquences finies d'éléments d'un vocabulaire fixé. Néanmoins, il apparaît judicieux de ne pas se contenter d'un telle définition pour plusieurs raisons : d'une part, certains travaux ultérieurs (notamment celui de KANAZAWA [60]) montreront que l'utilisation, dans le modèle de GOLD, de structures plus complexes que les phrases est tout aussi cohérente et surtout très utile. D'autre part, sur le plan linguistique, on peut envisager différents niveaux de représentation de la phrase, plus ou moins riches en informations.

Il n'y a donc aucune raison de limiter le modèle d'apprentissage à cette description naïve du langage. C'est pourquoi nous utiliserons dans cette partie le terme abstrait *objets* pour désigner les éléments d'un langage, quelle qu'en soit la forme. Selon les cas, ces objets pourront désigner de simples séquences de mots (phrases), mais aussi des structures spécifiques plus ou moins complexes (voir partie 3.2). Les langages sont donc simplement définis comme des ensembles (potentiellement infinis) d'objets. En pratique, on imposera souvent aux langages la contrainte supplémentaire d'être récursifs (ce qui sera précisé explicitement dans ce chapitre).

2.2.1.2 Apprenant et apprenabilité

Tous les *ensembles* dont il est question ci-dessous peuvent être infinis, sauf précision contraire explicite. Par analogie avec l'apprentissage humain, on parlera *d'apprenant*[3] pour désigner le "mécanisme" dont la tâche est de fournir une représentation grammaticale d'un langage décrit seulement par les objets qui le composent. Dans le cas de l'apprentissage par la machine, l'apprenant est une *fonction d'apprentissage*, et si cette fonction est calculable on parle aussi d'*algorithme d'apprentissage* (voir définition ci-dessous).

Une *classe de langages* est un ensemble de langages fixé, parfois aussi appelé *famille de langages* (par exemple dans [61]). Généralement il s'agit d'un ensemble de langages partageant une propriété particulière. Dans la suite on utilisera beaucoup les termes *apprenable* et *apprenabilité* : l'*apprenabilité* d'une classe de langages désigne son aptitude à être *apprise* selon la définition 2.1 ci-dessous. On trouve aussi dans la littérature différents termes désignant le caractère apprenable d'une classe de langages : *inférable, identifiable [à la limite]* ou *acquiérable* (le terme *acquisition* fait cependant plus souvent référence à l'apprentissage humain du langage).

2.2.2 Définition

Rappelons d'abord qu'un *système de grammaires* est spécifié par un triplet $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$, dans lequel $\mathcal{U}$ est un ensemble d'objets, $\mathcal{G}$ un ensemble de grammaires et $\mathcal{M}$ une fonction qui associe à chaque grammaire de $\mathcal{G}$ un sous-ensemble de $\mathcal{U}$ (définition 1.13).

Dans un système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$, une *fonction d'apprentissage* est une fonction partielle ϕ qui associe à des séquences finies non-vides d'objets de $\mathcal{U}$ des grammaire de $\mathcal{G}$:

[3] En anglais *learner*.

$$\phi : \bigcup_{k \geq 1} \mathcal{U}^k \mapsto \mathcal{G} \ ,$$

où $\mathcal{U}^k$ désigne l'ensemble des séquences d'éléments de $\mathcal{U}$ de longueur k.

Dans le modèle de GOLD [50], une séquence infinie d'objets est présentée à l'apprenant, qui doit alors "deviner" le langage ainsi énuméré. À chaque nouvel exemple, l'apprenant doit proposer une hypothèse sous forme d'une grammaire. Il y a *convergence* si au bout d'un nombre fini d'exemples l'hypothèse de l'apprenant n'est plus modifiée. Si de plus cette dernière grammaire correspond au langage correct, alors l'apprentissage est réalisé avec succès. Ce critère de succès est décrit formellement dans la définition suivante :

Définition 2.1 (Identification à la limite)**.** Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires, ϕ une fonction d'apprentissage et $L \subseteq \mathcal{U}$ un langage. Soit $\langle a_0, a_1, a_2, \ldots \rangle$ une séquence infinie d'objets de $\mathcal{U}$, telle que $a \in \{ a_i \mid i \in \mathbb{N} \}$ si et seulement si $a \in L$[4].

ϕ ***converge*** vers G s'il existe $n \in \mathbb{N}$ tel que pour tout $i \geq n$ $G_i = \phi(\langle a_1, a_2, \ldots, a_i \rangle)$ est définie et $G_i = G$.

ϕ ***apprend*** un langage L si, pour toute énumération de L, ϕ converge vers une grammaire G telle que $\mathcal{M}(G) = L$.

Une classe de langages $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ est dite ***apprenable*** s'il existe une fonction d'apprentissage ϕ telle que ϕ apprend L pour tout langage $L \in \mathcal{L}$.

On voit dans cette définition que la séquence d'exemples a quelques caractéristiques notables :

- Elle ne contient que des exemples *positifs*, c'est-à-dire des éléments du langage. La fonction d'apprentissage n'a donc aucune information extérieure au langage, ce qui constitue la principale difficulté de cette forme d'apprentissage.
- La séquence d'exemples est supposée ne comporter aucune erreur[5].
- La séquence d'exemples est une *énumération* du langage : tous les objets du langage doivent obligatoirement y apparaître. Ceci est possible y compris pour les langages infinis, parce que la séquence d'exemples est infinie.
- Les exemples peuvent apparaître dans un ordre quelconque dans la séquence, et éventuellement plusieurs fois (ce qui permet notamment d'énumérer indéfiniment un langage fini). Le modèle n'impose pas que la fonction d'apprentissage donne le même résultat pour deux séquences différentes contenant un même ensemble d'exemples.
- La séquence n'est pas vide, ce qui suppose que le langage non plus. On s'intéressera uniquement aux langages non vides, le langage vide ne présentant strictement aucun intérêt à apprendre[6].

Dans ce modèle, la convergence d'une fonction d'apprentissage n'a d'intérêt que si elle s'applique à un ensemble de langages, et non à un seul langage. En effet, d'après la définition ci-dessus, on voit

[4] Cette séquence est donc une *énumération* de L.

[5] Ce qui limite les applications potentielles : ce modèle est par définition inadapté aux données bruitées.

[6] Si nécessaire, le langage vide peut aisément faire l'objet d'un cas particulier, reconnaissable précisément à son énumération vide.

bien que ce dernier cas est trivial : il existe toujours une fonction capable d'apprendre un seul langage, puisqu'il suffit qu'elle soit définie de manière à toujours renvoyer la même grammaire (une grammaire du langage en question). Intuitivement, plus la classe de langages est grande, plus il est difficile de reconnaître un langage précis dans cette classe.

2.2.2.1 Apprenabilité des classes de grammaires

On parlera parfois dans la suite de l'apprenabilité de classes de *grammaires* : il s'agit d'une simplification pour parler des classes de langages correspondant aux grammaires de cette classe. Dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$, une classe de grammaires $\mathcal{G}_0 \subseteq \mathcal{G}$ est dite apprenable si et seulement si la classe de langages

$$\mathcal{L} = \{\ L \subseteq \mathcal{U} \mid \text{il existe une grammaire } G \in \mathcal{G}_0 \text{ telle que } \mathcal{M}(G) = L\ \}$$

est apprenable.

Lorsqu'on s'intéresse à l'apprenabilité de grammaires, il ne faut pas oublier que le succès de l'apprentissage est défini comme la convergence vers *une* grammaire représentant le langage correct. Sauf cas particulier, il peut y avoir plusieurs grammaires ayant le même langage dans une classe $\mathcal{G}_0$. Par conséquent l'apprenabilité de $\mathcal{G}_0$ ne permet pas d'affirmer que pour toute grammaire $G \in \mathcal{G}_0$ et toute énumération du langage de celle-ci la fonction d'apprentissage converge vers G.

2.2.2.2 Apprentissage effectif / non effectif

Le fait qu'une fonction d'apprentissage ne soit pas nécessairement une fonction totale signifie que l'apprenant n'est pas "obligé de répondre" à chaque étape de l'énumération. Cependant il est évident qu'il ne peut y avoir convergence si la fonction d'apprentissage ne répond jamais. La fonction d'apprentissage peut même ne pas être calculable (autrement dit la fonction est non récursive), on parle alors d'*apprenabilité non effective* :

Définition 2.2 (Apprenabilité effective)**.** Une classe de langages $\mathcal{L}$ est *non effectivement apprenable* s'il existe une fonction d'apprentissage qui apprend $\mathcal{L}$. Une classe de langages $\mathcal{L}$ n'est pas *non effectivement apprenable* s'il n'existe pas de telle fonction.

Une classe de langages $\mathcal{L}$ est *(effectivement) apprenable* s'il existe une fonction *calculable* d'apprentissage qui apprend $\mathcal{L}$. Une classe de langages $\mathcal{L}$ n'est pas *effectivement apprenable* s'il n'existe pas de telle fonction.

On voit bien ici que l'apprenabilité effective implique l'apprenabilité non effective. Par la suite on s'intéressera uniquement à l'apprentissage effectif[7], bien que les résultats négatifs d'apprenabilité présentés soient en fait des résultats négatifs d'apprenabilité non effective (c'est-à-dire que dans ce cas il n'existe aucune fonction d'apprentissage, pas seulement de fonction d'apprentissage calculable).

[7] Lorsque cela ne présente pas d'ambiguité, nous parlerons donc simplement *d'apprenabilité* au lieu *d'apprenabilité effective*.Voir les travaux d'OSHERSON, STOB et WEINSTEIN [83] sur l'apprenabilité non effective.

Lorsque la fonction d'apprentissage est calculable on parle aussi d'*algorithme d'apprentissage*. L'ensemble de toutes les classes de langages effectivement apprenables est généralement noté *TxtEx* dans la littérature.

2.2.2.3 Apprentissage par énumération

L'apprentissage par énumération est une méthode générale qui consiste à faire une recherche systématique dans l'ensemble des grammaires possibles, jusqu'à en trouver une qui engendre les exemples vus jusqu'alors et la renvoyer comme hypothèse. Cette méthode ne permet de réaliser l'apprentissage correct dans le modèle de GOLD que si les conditions suivantes sont remplies :

- Il existe un moyen d'énumérer l'ensemble des grammaires correpondant aux langages possibles.
- Il existe un moyen de vérifier qu'un objet appartient au langage d'une grammaire proposée comme hypothèse.
- Pour toute hypothèse fausse proposée (c'est-à-dire ne correspondant pas exactement au langage à deviner), il y a dans la suite de la séquence d'objets un ensemble fini d'objets qui montre cette erreur.

L'erreur qui consiste seulement pour la grammaire hypothèse à ne pas générer tout le langage est aisément repérable, puisqu'un objet appartenant au langage cible mais pas à celui de la grammaire apparaîtra tôt ou tard dans la séquence. Mais la dernière condition doit surtout servir à éviter la surgénéralisation, c'est-à-dire le fait que le langage de la grammaire hypothèse soit un sur-ensemble strict du langage cible. Il faut par conséquent éviter de proposer un langage qui soit un sur-ensemble strict du langage cible. Ceci est possible par exemple si l'énumération peut être faite selon un ordre qui garantit que si une grammaire G' apparaît après G, alors le langage de G' n'est pas inclus strictement dans celui de G.

L'énumération n'est pas une méthode d'apprentissage universelle, parce qu'il existe des classes de langages apprenables qui ne sont pas apprenables par énumération. COSTA FLORÊNCIO donne l'exemple suivant [35] :

Exemple 2.1 (Classe de langages apprenable mais non apprenable par énumération)**.** Soit le système de grammaires $\langle \mathbb{N}, \mathbb{N}, f \rangle$, où f est une fonction qui à tout $n \in \mathbb{N}$ associe le langage $L_n = \{n, n+1, n+2, \ldots\}$ (les grammaires sont de simples indices).

Soit $\mathcal{L}$ la classe constituée de tous les L_n : $\mathcal{L} = \{\, L_n \mid n \in \mathbb{N} \,\}$.

Il existe une fonction ϕ qui apprend $\mathcal{L}$: pour toute séquence finie d'éléments de $\mathbb{N}$, ϕ renvoie le minimum parmi ces nombres.

Mais $\mathcal{L}$ n'est pas apprenable par énumération, parce qu'une énumération des grammaires par ordre croissant provoquerait une surgénéralisation, tandis qu'une énumération par ordre décroissant est clairement impossible.

On parle néanmoins parfois d'*algorithme d'apprentissage universel* pour désigner un algorithme qui effectue une telle forme de recherche systématique, par opposition aux algorithmes utilisant des propriétés spécifiques de la classe de langages pour générer une hypothèse seulement à partir des exemples. Il

s'agit là d'un abus de langage, puisque la classe de tous les langages apprenables n'est pas *uniformément apprenable* (voir la partie 2.4.1).

2.2.3 Conséquences immédiates

Dans cette partie nous proposons de décrire les propriétés qui découlent directement de la définition du modèle. La plupart de ces propriétés ont été découvertes par GOLD. La première propriété ci-dessous est triviale :

Proposition 2.1. *Soit $\mathcal{L}$ une classe de langage apprenable. Pour toute classe de langages $\mathcal{L}'$ telle que $\mathcal{L}' \subseteq \mathcal{L}$, $\mathcal{L}'$ est aussi apprenable.*

Par contraposée, si une classe n'est pas apprenable alors toute classe qui la contient (super-classe) ne l'est pas non plus.

2.2.3.1 Point limite

Cette définition de l'identification à la limite a d'importantes conséquences immédiates, démontrées par GOLD dans [50]. La première est un résultat positif pour les langages finis.

Proposition 2.2 (GOLD)**.** *La classe des langages de cardinalité finie est apprenable.*

Il suffit dans ce cas que l'apprenant ajoute un par un les exemples présentés à la grammaire hypothèse. Comme le langage est fini et que tous ses éléments sont présents dans la séquence d'exemples, la fonction d'apprentissage converge dès que le langage a été énuméré en totalité[8].

En revanche la seconde conséquence de la définition du modèle est un résultat négatif : toute classe de langages contenant tous les langages finis et au moins un langage infini n'est pas apprenable. Il s'agit en fait d'une conséquence du théorème plus général concernant l'existence d'un *point limite*[9] :

Théorème 2.1 (Point limite)**.** *Un langage $L_\infty \in \mathcal{L}$ est un point limite pour une classe de langages $\mathcal{L}$ s'il existe une séquence infinie $\langle L_0, L_1, L_2, \ldots \rangle$ de langages dans $\mathcal{L}$ telle que*

$$L_i \subset L_{i+1} \text{ pour tout } i \geq 0 \quad \text{et} \quad L_\infty = \bigcup_{n \in \mathbb{N}} L_n.$$

Si la classe de langages $\mathcal{L}$ a un point limite, alors $\mathcal{L}$ n'est pas apprenable.

On peut grossièrement résumer la preuve de ce théorème de la manière suivante : supposons qu'il existe une fonction d'aprentissage qui apprend $\mathcal{L}$ et qu'on lui présente des éléments du langage L_1, puis des éléments du langage L_2, et ainsi de suite à l'infini. A chaque étape n, la fonction d'apprentissage doit choisir entre le langage L_n et le langage L_∞ et ne dispose d'aucun indice pour cela : si elle choisit L_n (méthode prudente), elle n'atteindra jamais L_∞, donc ne converge pas vers le bon langage dans le cas

[8] La manière de construire une telle grammaire dépend de la relation qui relie les grammaires aux langages. Dans le cas des langages récursifs, on peut voir cette propriété comme une conséquence évidente de l'élasticité finie (voir proposition 2.3).

[9] Le terme *point limite* étant dû à KAPUR [61].

d'une énumération de L_∞. A l'inverse, si à l'étape n la fonction renvoie comme hypothèse le langage L_∞ (généralisation), alors elle ne peut pas apprendre L_n. Par conséquent il est impossible d'apprendre une classe contenant un tel ensemble de langages.

Un point limite est en fait un cas particulier de point d'accumulation (voir définition 2.5). On peut donc se baser sur la démonstration de la proposition 2.6 pour obtenir une preuve formelle du théorème du point limite.

Exemple 2.2 (Point limite pour les langages réguliers)**.** Pour tout $n \geq 1$ on définit $L_n = \{ x^i \mid i \leq n \}$ comme le langage des chaînes de x de longueur inférieure ou égale à n, avec $x \in \Sigma$. Soit $L_\infty = x^*$ le langage de toutes les chaînes de x, et $\mathcal{L}$ une classe de langages contenant tous les L_n ($n \in \mathbb{N}$) ainsi que L_∞.

On voit que $L_{n+1} = L_n \cup \{x^{n+1}\}$ pour tout n, donc $L_n \subset L_{n+1}$. De plus tout langage L_n est inclus (strictement) dans L_∞, qui est par conséquent un point limite pour $\mathcal{L}$. Donc $\mathcal{L}$ n'est pas apprenable.

Les langages $\{ L_n \mid n \in \mathbb{N} \}$ et L_∞ définis dans l'exemple ci-dessus sont tous réguliers, ce qui amène à un corollaire important du théorème 2.1 : ces langages étant réguliers, toutes les classes de la hiérarchie de CHOMSKY les contiennent, et ne sont par conséquent pas apprenables. Le fait que même la classe la plus simple de la hiérarchie de CHOMSKY, celle des langages réguliers, ne soit pas apprenable dans le modèle de GOLD a longtemps constitué un obstacle majeur au développement du modèle. Le modèle d'identification à la limite à partir d'exemples positifs était alors considéré comme inadapté, car trop contraignant. GOLD lui-même notait : "*Cependant, les résultats présentés dans la dernière section montrent que seule la classe de langages la plus triviale considérée*[10] *est apprenable à partir d'exemples positifs[..].*"

Il est souvent question dans la littérature du domaine de classes de langages *triviales* (ou *non triviales*) à apprendre. Étant donné les principales caractéristiques du modèle de GOLD vues ci-dessus, on peut considérer de manière informelle qu'une classe de langages est *non triviale* (pour l'apprentissage) si elle comporte au moins un nombre infinis de langages, dont certains sont infinis. Il est clair que si ces conditions ne sont pas remplies la classe est apprenable[11], et accessoirement ne présente pas grand intérêt du point de vue de sa structure.

2.3 Caractérisation des classes de langages apprenables

L'étude de l'apprenabilité dans le modèle de GOLD peut bien sûr être faite au cas par cas, en observant un système de grammaires particulier pour y découvrir un algorithme d'apprentissage, ou au contraire une propriété qui montre la non apprenabilité (comme un point limite). Il est cependant plus avantageux de rechercher les propriétés intrinsèques du modèle qui rendent possible ou impossible l'apprentissage, aussi bien pour des raisons pratiques (appliquer un théorème général est plus rapide que de

[10] Il s'agit de la classe des langages de cardinalité finie.

[11] Le fait qu'une classe ne contenant qu'un nombre fini de langages est apprenable peut être vu comme une conséquence de l'élasticité finie (voir proposition 2.3).

le démontrer entièrement sur plusieurs cas particuliers), que théoriques (comprendre les mécanismes du modèle indépendamment d'une de ses instances particulières).

La recherche de caractéristiques qui distinguent les classes de langages apprenables des non apprenables a donc été le principal objet des travaux sur le modèle de GOLD. On s'intéresse surtout à dégager des conditions nécessaires et/ou suffisantes qui garantissent l'apprenabilité.

2.3.1 Séquences de verrouillage

BLUM et BLUM montrent une première propriété[12] des langages apprenables dans [20], à savoir l'existence dans toute énumération d'un tel langage d'une *séquence de verrouillage*.

Proposition 2.3 (Séquence de verrouillage)**.** *Si L est un langage tel qu'il existe une fonction d'apprentissage ϕ qui converge vers L pour toute énumération de L, alors il existe une séquence finie $\langle a_0, \ldots, a_n\rangle$ telle que*

- $\{a_0, \ldots, a_n\} \subseteq L$,
- *et pour toute séquence finie $\langle b_0, \ldots, b_m\rangle$ telle que $\{b_0, \ldots, b_m\} \subseteq L$ on a $\phi(\langle a_0, \ldots, a_n\rangle) = \phi(\langle a_0, \ldots, a_n, b_0, \ldots, b_m\rangle)$.*

Une telle séquence est appelée une séquence de verrouillage *de L pour ϕ.*

Démonstration. Supposons qu'il n'existe pas de séquence de verrouillage : pour toute séquence finie $\langle a_0, \ldots, a_n\rangle$ telle que $\{a_0, \ldots, a_n\} \subseteq L$ il existe une séquence finie $\langle b_0, \ldots, b_m\rangle$ telle que

$$\phi(\langle a_0, \ldots, a_n\rangle) \neq \phi(\langle a_0, \ldots, a_n, b_0, \ldots, b_m\rangle).$$

Soit $\langle s_i\rangle_{i\in\mathbb{N}}$ une énumération infinie de L. On construit par induction une seconde séquence infinie $\langle u_i\rangle_{i\in\mathbb{N}}$ qui énumère L, pour laquelle ϕ ne converge pas (ce qui constitue une contradiction) :

- Étape 0. Soit $u_0 = s_0$, et l'indice i_0 est initialisé à 0.
- Étape $n+1$. Les objets $u_0, \ldots, u_{i_n}$ ont été définis dans les étapes précédentes. Par construction $\{u_0, \ldots, u_{i_n}\} \subseteq L$, donc d'après notre hypothèse il existe une séquence $\langle b_0, \ldots, b_m\rangle$ telle que $b_j \in L$ pour tout j et $\phi(\langle u_0, \ldots, u_{i_n}\rangle) \neq \phi(\langle u_0, \ldots, u_{i_n}, b_0, \ldots, b_m\rangle)$. Soit $u_{i_n+1+j} = b_j$ pour tout $0 \leq j \leq m$, et $u_{i_n+m+2} = s_{n+2}$. L'indice i_{n+1} est fixé à $i_n + m + 2$.

La séquence $\langle u_i\rangle_{i\in\mathbb{N}}$ est une énumération de L puisque par construction $L = \{s_i | i \in \mathbb{N}\} \subseteq \{u_i | i \in \mathbb{N}\} \subseteq L$. Or ϕ ne peut converger sur cette séquence car pour tout n $\phi(\langle u_0, \ldots, u_{i_n}\rangle) \neq \phi(\langle u_0, \ldots, u_{i_{n+1}-1}\rangle)$. □

Remarque : Si $\langle a_0, \ldots, a_n\rangle$ est une séquence de verrouillage de L pour ϕ, alors toute séquence finie $\langle a_0, \ldots, a_n, b_0, \ldots, b_m\rangle$ d'éléments de L l'est aussi.

Corollaire 2.1 (BLUM et BLUM)**.** *Soit ϕ une fonction qui apprend la classe de langages $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M}\rangle$. Pour tout $L \in \mathcal{L}$, il existe une séquence de verrouillage $\langle a_0, \ldots, a_n\rangle$ de L pour ϕ telle que $\mathcal{M}(\phi(\langle a_0, \ldots, a_n\rangle)) = L$.*

[12] La formulation de la preuve de la proposition 2.3 proposée ci-dessous est tirée de [60].

Démonstration. ϕ apprend $\mathcal{L}$, donc ϕ converge pour toute énumération de L. D'après la proposition 2.3, il existe une séquence de verrouillage $\langle a_0, \ldots, a_n \rangle$ de L pour ϕ. Soit $\langle b_i \rangle_{i \in \mathbb{N}}$ une énumération infinie de L telle que $a_i = b_i$ pour tout $i \leq n$: cette énumération de L existe puisque $\{a_0, \ldots, a_n\} \subseteq L$. Selon la propriété des séquences de verrouillage (proposition 2.3), pour tout $i \geq n$: $\phi(\langle b_0, \ldots, b_i \rangle) = \phi(\langle a_0, \ldots, a_n \rangle)$. Comme ϕ apprend $\mathcal{L}$ et que $L \in \mathcal{L}$, ϕ converge vers une grammaire G telle que $\mathcal{M}(G) = L$, par conséquent $\mathcal{M}(\phi(\langle a_0, \ldots, a_n \rangle)) = L$. □

L'existence d'une séquence de verrouillage dans une énumération est donc une condition nécessaire pour la convergence de la fonction d'apprentissage vers le langage correct. Cette condition a pour inconvénient de porter sur les énumérations des langages, ce qui la rend relativement difficile à utiliser.

2.3.2 Ensembles révélateurs (théorème d'ANGLUIN)

Dans le cadre de l'apprentissage à partir d'exemples positifs, on a vu que le principal obstacle à franchir est d'éviter la surgénéralisation. Les résultats présentés ci-dessous vont dans le sens d'une recherche de conditions qui garantissent la possibilité de ne pas faire de surgénéralisation. La première étape de cette recherche a été accomplie par ANGLUIN, à travers ce qu'elle nomme les *ensembles révélateurs* (*tell-tale sets*, [5]). Elle obtient, à l'aide de son théorème présenté ci-dessous, le premier résultat important d'apprenabilité sur les *langages de motifs* (*pattern languages*) [4].

On peut décrire sommairement le principe des ensembles révélateurs de la façon suivante : il faut que chaque langage dans la classe ait une sorte de "signature" qui le distingue de tous les langages dont il est un sur-ensemble strict. Ainsi, lorsque cette signature apparaît dans la séquence d'exemples, on peut proposer le langage en question sans risque de surgénéralisation.

2.3.2.1 Définitions

Définition 2.3 (Ensemble révélateur (ANGLUIN))**.** Soit $\mathcal{L}$ une classe de langages. Un ensemble fini non vide[13] d'objets D est un ensemble révélateur du langage $L \in \mathcal{L}$ si $D \subseteq L$ et pour tout langage $L' \in \mathcal{L}$:

$$L' \subset L \Rightarrow D \nsubseteq L'$$

Il est utile de noter que si T est un ensemble révélateur de L, alors tout ensemble fini T' tel que $T \subseteq T' \subseteq L$ est aussi un ensemble révélateur de L.

Proposition 2.4. *Soit ϕ une fonction qui apprend la classe de langages $\mathcal{L}$. Si $\langle a_0, \ldots, a_n \rangle$ est une séquence de verrouillage du langage $L \in \mathcal{L}$ pour ϕ, alors $\{a_0, \ldots, a_n\}$ est un ensemble révélateur de L dans $\mathcal{L}$.*

Démonstration. (Par l'absurde) Supposons que $\{a_0, \ldots, a_n\}$ ne soit pas un ensemble révélateur de L. Alors il existe un langage $L' \in \mathcal{L}$ tel que $\{a_0, \ldots, a_n\} \subseteq L' \subset L$. Soit $\langle b_i \rangle_{i \in \mathbb{N}}$ une énumération infinie

[13] La définition d'origine n'impose pas cette contrainte. Elle est cependant naturelle dans la mesure où on s'intéresse ici à des langages non vides. En effet, il existe toujours un ensemble révélateur D non vide si l'ensemble vide est révélateur de L, puisqu'alors il n'existe aucun $L' \subset L$. On peut donc prendre n'importe quel $x \in L$ comme unique élément de D.

de L' telle que $a_i = b_i$ pour tout $i \leq n$ (cette énumération existe, puisque $\{a_0, \ldots, a_n\} \subseteq L'$). Comme $\{a_0, \ldots, a_n\}$ est une séquence de verrouillage de L pour ϕ, d'après la proposition 2.3 $\phi(\langle b_0, \ldots b_i \rangle) = L \neq L'$ pour tout $i \geq n$. Or $L' \in \mathcal{L}$ et $\langle b_i \rangle_{i \in \mathbb{N}}$ est une énumération de L', donc ϕ n'apprend pas la classe $\mathcal{L}$, ce qui constitue une contradiction. □

Le théorème central d'ANGLUIN présenté ci-dessous s'applique seulement à des *familles indexées de langages récursifs*[14]. Cette restriction reste très raisonnable : elle impose simplement d'une part que la classe de langages soit récursivement énumérable, et d'autre part que les langages soient récursifs (en d'autres termes que le problème de l'appartenance soit décidable). Nous proposons une définition de "famille indexée" utilisant l'énumérabilité des grammaires représentant les langages (comme dans [60], [35]), plutôt que l'énumérabilité des langages eux-mêmes (comme c'est le cas dans la définition d'origine [5], les grammaires étant alors de simples indices). Les deux versions sont équivalentes. Rappelons aussi qu'on ne s'intéresse ici qu'aux langages non vides.

Définition 2.4 (Famille indexée de langages)**.** Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires et $\mathcal{G}_0 \subset \mathcal{G}$ un ensemble récursivement énumérable de grammaires.

La classe de langages $\mathcal{L} = \{ L = \mathcal{M}(G) \,|\, G \in \mathcal{G}_0 \}$ est une *famille indexée de langages récursifs* s'il existe une fonction calculable $f : \mathcal{G} \times \mathcal{U} \mapsto \{0, 1\}$ telle que $f(G, x) = 1$ si $x \in \mathcal{M}(G)$ et $f(G, x) = 0$ sinon, pour tout $G \in \mathcal{G}_0$ et tout $x \in \mathcal{U}$.

2.3.2.2 Algorithmes d'énumération d'un ensemble fini

Le théorème central d'ANGLUIN, présenté ci-dessous, repose sur l'existence d'un *algorithme (calculable) d'énumération d'un ensemble fini*. Comme on peut le constater dans la preuve du théorème, il n'est pas nécessaire[15] que cet algorithme "signale la fin de l'énumération" (formulation utilisée par KAPUR dans [61]). Dans [37], DE JONGH et KANAZAWA en donnent une version plus formelle : l'existence de cet algorithme qui n'a pas besoin de signaler la fin de l'énumération équivaut à l'existence d'une fonction *récursive à la limite*, c'est-à-dire une fonction partielle $\varphi : \mathbb{N} \mapsto \mathbb{N}$ telle qu'il existe une fonction récursive totale $F : \mathbb{N}^2 \mapsto \mathbb{N}$ telle que $\varphi(x) \simeq \lim_{n \to \infty} F(x, n)$ (avec $\lim_{n \to \infty} f(x, n)$ définie et égale à x si et seulement si $\exists m \forall n \geq m, f(n) \simeq x$). On voit que la classe des fonctions récursives partielles est incluse dans celle des fonctions récursives à la limite. Voir [37] pour plus de détails.

Nous proposons ici de considérer qu'un algorithme d'énumération d'un ensemble fini est un algorithme prenant en entrée un indice $n \in \mathbb{N}$ et écrivant en sortie les n premiers éléments de l'énumération. Afin de garantir à la fois que l'algorithme termine et que les valeurs en sortie sont définies quel que soit $n \in \mathbb{N}$, l'algorithme pourra renvoyer des éléments déjà sortis de l'ensemble fini. Cet algorithme effectue donc simplement une énumération infinie d'un ensemble fini, avec pour contrainte de toujours prendre le même ordre d'énumération : si pour un n donné le $i^{\text{ème}}$ élément est x, alors le $i^{\text{ème}}$ élément est toujours x quel que soit $n \geq m$.

[14] On parlera simplement de "famille indexée".

[15] Cela est même impossible pour obtenir l'équivalence de ce théorème.

2.3.2.3 Théorème d'ANGLUIN

Théorème 2.2 (ANGLUIN)**.** *Soit $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ une famille indexée de langages récursifs dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$:*

$$\mathcal{L} = \{ \mathcal{M}(G) \mid G \in \{G_0, G_1, G_2, \ldots\} \}$$

Il existe une fonction ϕ qui apprend $\mathcal{L}$ si et seulement s'il existe un algorithme calculable qui, pour tout indice I tel que $L_I = \mathcal{M}(G_I) \in \mathcal{L}$ énumère récursivement un ensemble révélateur D_I de L_I.

Démonstration. $\Rightarrow$. Supposons qu'il existe une fonction ϕ qui apprend $\mathcal{L}$. $\mathcal{L}$ est une famille indexée de langages récursifs, donc étant donné un indice I il est possible d'énumérer le langage $L_I = \mathcal{M}(G_i)$, et aussi d'énumérer l'ensemble des séquences du langage. L'algorithme 2.1 effectue une recherche systématique d'une séquence de verrouillage de L_I, en utilisant la fonction $EnumSequenceLangage(L, n)$ qui renvoie la $n^{\text{ème}}$ séquence de L.

EnumRevelateur(I**,**n**)**
- $i \leftarrow 0$
- $S \leftarrow EnumSequenceLangage(G_I, i)$
- $R \leftarrow S$
- $j \leftarrow 0$
- Soit x un élément arbitraire de $\mathcal{M}(G_I)$
- **Tant que (vrai) faire**
  - $S' \leftarrow EnumSequenceLangage(G_I, j)$
  - **Si** ($\phi(S) = \phi(S \bullet S')$) **Alors**
    - $j \leftarrow j + 1$
    - $R \leftarrow R \bullet \langle x \rangle$
  - **Sinon**
    - $j \leftarrow 0$
    - $i \leftarrow i + 1$
    - $S \leftarrow EnumSequenceLangage(G_I, i)$
    - $R \leftarrow R \bullet S$
  - **Fin Si**
  - **Si** (R contient plus de n éléments) **Alors**
    - **Renvoyer** les n premiers éléments de la séquence R
  - **Fin Si**
- **Fin Tant Que**

ALG. 2.1 – Énumération d'un ensemble révélateur à partir d'une fonction d'apprentissage

D'après la proposition 2.3, une séquence de verrouillage de L_I pour ϕ existe, donc il existe un i pour lequel $S = EnumSequenceLangage(G_I, i)$ est une séquence de verrouillage. Ce i sera nécessairement atteint, pour n suffisamment grand.

À partir de cette étape, la condition $\phi(S) = \phi(S \bullet S')$ sera toujours vraie (par définition de la séquence de verrouillage) donc l'algorithme bouclera jusqu'à ce que le nombre de x ajoutés à R soit suffisant pour que R contienne plus de n éléments (les x sont ajoutés uniquement pour compléter la séquence jusqu'à n). Ainsi, quel que soit n, le nombre d'éléments distincts dans R est borné (autrement dit la fonction énumère bien un ensemble fini). Les éléments de la dernière séquence S forment un ensemble révélateur (d'après la proposition 2.4). Par conséquent R, la sortie de la fonction, est composée d'éléments de L_I contenant un ensemble révélateur, ce qui constitue bien un ensemble révélateur de L_I pour ϕ.

$\Leftarrow$. Supposons qu'il existe un algorithme $EnumRevel(I, n)$ qui, pour tout $L_I \in \mathcal{L}$ énumère récursivement un ensemble révélateur D_I de L_I. Alors l'algorithme 2.2 apprend[16] la classe $\mathcal{L}$.

ϕ**(**$\langle a_0, \ldots, a_n \rangle$**)**
  $i \leftarrow 0$
  $E \leftarrow EnumRevel(i, n)$
  Tant que ($i \leq n$ **et non**($\{a_0, \ldots, a_n\} \subseteq \mathcal{M}(G_i)$ **et** $E \subseteq \{a_0, \ldots, a_n\}$)) **faire**
    $i \leftarrow i + 1$
    $E \leftarrow EnumRevel(i, n)$
  Fin Tant Que
  Renvoyer G_i

ALG. 2.2 – Apprentissage d'une famille indexée à partir de l'ensemble révélateur

Soit $L \in \mathcal{L}$ et $\langle a_i \rangle_{i \in \mathbb{N}}$ une énumération de L. Dans la suite on note L_i le langage $\mathcal{M}(G_i)$ (pour tout i). Soit m le plus petit entier tel que $L_m = L$. Soit N le plus petit entier tel que

i. pour tout $i < m$ tel que $L_m \setminus L_i \neq \emptyset$, il existe $x \in L_m \setminus L_i$ tel que $x \in \{a_0, \ldots, a_N\}$,

ii. pour tout $i < m$ tel que $L_m \subset L_i$, $EnumRevel(i, N)$ contient un ensemble révélateur complet de L_i,

iii. $EnumRevel(m, N) \subseteq \{a_0, \ldots, a_N\}$, et

iv. $N \geq m$.

Soit $n \geq N$. Montrons que pour tout $i < m$ la condition du *Tant que* est vraie, donc la boucle continue. $i \leq n$ car $i < m \leq N \leq n$ (iv). Si $L_m \setminus L_i \neq \emptyset$, alors il existe $x \notin L_i$ tel que $x \in \{a_0, \ldots, a_n\}$ (i), donc la condition est vraie. Sinon, si $L_m \subset Li$, alors $E = EnumRevel(i, n)$ contient un ensemble révélateur complet D_i de L_i (ii), donc d'après la définition 2.3 $D_i \subseteq E \not\subseteq L_m$, ce qui implique $E \not\subseteq \{a_0, \ldots, a_n\}$. On montre ensuite que si $i = m$ la condition du *Tant que* est fausse : $\{a_0, \ldots, a_n\} \subseteq \mathcal{M}(G_m)$, et $E = EnumRevel(m, n) \subseteq \{a_0, \ldots a_n\}$ (iii). Ainsi dans tous les cas la boucle continue tant que $i < m$, et s'arrête lorsque $i = m$. Par conséquent ϕ renvoie toujours G_m telle que $L = \mathcal{M}(G_m)$, pour tout $n \geq N$, pour toute énumération de L et pour tout $L \in \mathcal{L}$.

□

[16] Dans la version simple proposée ici, on ne cherche pas à renvoyer une grammaire consistante avec les données (avant le point de convergence bien sûr). En particulier, si c'est la condition $i \leq n$ qui n'est plus vérifiée la grammaire G_i est renvoyée arbitrairement : cette grammaire a de grandes chances de ne pas correspondre au langage énuméré.

La nécessité de l'existence d'un algorithme énumérateur des ensembles révélateurs est un aspect très important du théorème d'ANGLUIN ci-dessus. En effet, la simple existence d'un ensemble révélateur pour chaque langage de la classe n'est pas une condition suffisante pour que la classe de langages soit apprenable [5] (Celle-ci est en fait une condition nécessaire et suffisante pour l'apprenabilité *non effective* [83]). KAPUR proposera une condition plus forte basée sur *les ensembles de test* (décrite dans la partie 2.4.1).

2.3.3 Points d'accumulation

Le théorème d'ANGLUIN donne une caractérisation des classes de langages apprenables en termes d'existence d'un algorithme d'énumération. KAPUR propose une seconde équivalence, basée sur les caractéristiques structurelles des classes de langages [61] :

Définition 2.5 (Point d'accumulation)**.** Soit $\mathcal{L}$ une classe de langages. Un langage $L \in \mathcal{L}$ est un point d'accumulation s'il existe une séquence infinie d'ensembles $\langle S_i \rangle_{i\in\mathbb{N}}$ telle que

i. pour tout i, $S_i \subseteq S_{i+1}$,

ii. $\bigcup_{i\in\mathbb{N}} S_i = L$ et

iii. pour tout i, il existe $L' \in \mathcal{L}$ tel que $S_i \subseteq L'$ et $L' \subset L$.

Remarque : La définition du point d'accumulation fournie par KAPUR dans [61] stipule que les ensembles S_i doivent être finis. Les deux définitions sont équivalentes : Si une séquence $\langle S_i \rangle_{i\in\mathbb{N}}$ d'ensembles finis existe, il est évident que cette séquence est correcte aussi sans contrainte sur les ensembles. Si une séquence $\langle S_i \rangle_{i\in\mathbb{N}}$ d'ensembles (potentiellement infinis) existe, on montre qu'il existe une séquence $\langle S'_i \rangle_{i\in\mathbb{N}}$ d'ensembles finis qui remplit les conditions. Soit $\langle a_i \rangle_{i\in\mathbb{N}}$ une énumération de L, et $A_i = \{a_0, \ldots, a_i\}$ l'ensemble (fini) des i premiers éléments de cette énumération. On définit $S'_i = S_i \cap A_i$. Clairement, $S_i \subseteq S_{i+1}$ implique que $S_i \cap A_i \subseteq S_{i+1} \cap A_{i+1}$ (i). De même, si $S_i \subseteq L' \subset L$ alors $S'_i = S_i \cap A_i \subseteq L' \subset L$ (ii). Pour tout $x \in L$, il existe S_i et A_j tels que $x \in S_i$ et $x \in A_j$. Si $i \leq j$, alors $x \in S_j \cap A_j = S'_j$ car $S_i \subseteq S_j$. Sinon ($i > j$), $x \in S_i \cap A_i = S'_i$ car $A_j \subseteq A_i$. Par conséquent L est inclus dans l'union infinie des S'_i, et comme tous les S_i et A_i sont des sous-ensembles de L on a bien $L = \bigcup_{i\in\mathbb{N}} S'_i$ (iii).

Proposition 2.5. *Tout point limite est un point d'accumulation.*

Démonstration. Soit $L_1 \subset L_2 \subset \ldots$ une séquence infinie de langages telle que l'union de tous les L_i est L, avec $\{L, L_1, L_2, \ldots\} \subseteq \mathcal{L}$. L est un point limite de $\mathcal{L}$ (définition 2.1). Il suffit de prendre $S_i = L_i$ pour tout i pour montrer que L est aussi un point d'accumulation. □

Proposition 2.6 (KAPUR)**.** *Soit $\mathcal{L}$ une classe de langages. Un langage $L \in \mathcal{L}$ a un ensemble révélateur si et seulement si L n'est pas un point d'accumulation.*

Démonstration. $\Rightarrow$. Supposons que L est un point d'accumulation : il existe une séquence infinie $\langle S_i \rangle_{i \in \mathbb{N}}$ vérifiant les conditions de la définition 2.5. Quel que soit D un sous-ensemble fini de L, il existe $n \in \mathbb{N}$ tel que $D \subseteq S_n$, car $S_i \subseteq S_{i+1}$ pour tout i, et l'union de tous les S_i est L. Or d'après la condition (iii) il existe $L' \in \mathcal{L}$ tel que $D \subseteq S_n \subseteq L' \subset L$, donc D n'est pas un ensemble révélateur de L.

$\Leftarrow$. Supposons que L n'a pas d'ensemble révélateur. L est donc infini, sinon L serait son propre ensemble révélateur. Soit $\langle a_i \rangle_{i \in \mathbb{N}}$ une énumération de L, et soit $S_i = \{a_0, \ldots, a_i\}$ pour tout i. On a bien $S_i \subseteq S_{i+1}$ (condition (i)) et l'union de tous les S_i est L (condition (ii)). Comme L n'a pas d'ensemble révélateur, pour tout $D \subseteq L$ fini, il existe $L' \in \mathcal{L}$ tel que $L' \subset L$ et $D \subseteq L'$. Tous les S_i sont finis, donc pour tout i il existe $L' \in \mathcal{L}$ tel que $S_i \subseteq L'$ et $L' \subset L$ (condition (iii)). Par conséquent L est un point d'accumulation. □

L'existence d'un point limite est une condition suffisante pour la non apprenabilité (non effective). La condition du point d'accumulation est plus forte : son existence est une condition nécessaire et suffisante à la non apprenabilité (non effective).

Le corollaire ci-dessous résume les principaux résultats présentés dans cette partie.

Corollaire 2.2 (Récapitulatif)**.** *Avec $\mathcal{L}$ une famille indexée de langages récursifs :*

La classe $\mathcal{L}$ est apprenable
⇳ *(déf. 2.1)*
Il existe une fonction ϕ qui apprend la classe $\mathcal{L}$
⇳ *(prop. 2.2)*
$\forall L \in \mathcal{L}$ il est possible d'énumérer un ensemble révélateur de L
⇓
Tout langage $L \in \mathcal{L}$ a un ensemble révélateur
⇳ *(prop. 2.6)*
Aucun langage $L \in \mathcal{L}$ n'est un point d'accumulation pour $\mathcal{L}$
⇓ *(prop. 2.5)*
Aucun langage $L \in \mathcal{L}$ n'est un point limite pour $\mathcal{L}$

Il existe un point limite $L \in \mathcal{L}$
⇓ *(prop. 2.5)*
Il existe un point d'accumulation $L \in \mathcal{L}$
⇳ *(prop. 2.6)*
$\exists L \in \mathcal{L}$ qui n'a pas d'ensemble révélateur
⇓
$\exists L \in \mathcal{L}$ tel qu'il est impossible d'énumérer un ensemble révélateur de L
⇳ *(prop. 2.2)*
Il n'existe pas de fonction ϕ qui apprend la classe $\mathcal{L}$
⇳ *(déf. 2.1)*
La classe $\mathcal{L}$ n'est pas apprenable

2.4 L'élasticité finie

WRIGHT propose une condition suffisante nommée l'*élasticité finie* pour l'apprenabilité d'une classe de langages. Ce critère est introduit dans [109], puis corrigé par MOTOKI, SHINOHARA et WRIGHT dans [80] (car la définition d'origine contenait une erreur).

Définition 2.6 (Élasticité infinie (WRIGHT))**.** Dans un système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$, une classe de langages $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ a la propriété d'*élasticité infinie* s'il existe une séquence infinie $\langle a_i \rangle_{i \in \mathbb{N}}$ d'objets dans $\mathcal{U}$ et une séquence infinie $\langle L_i \rangle_{i \in \mathbb{N}^*}$ de langages dans $\mathcal{L}$ tels que pour tout $i \in \mathbb{N}^*$

$$a_i \notin L_i$$
et
$$\{a_0, \ldots, a_{i-1}\} \subseteq L_i$$

Définition 2.7 (Élasticité finie (WRIGHT))**.** Une classe de langages $\mathcal{L}$ a la propriété d'*élasticité finie* si elle n'a pas l'élasticité infinie.

Proposition 2.7 (WRIGHT)**.** *Si $\mathcal{L}$ a l'élasticité finie, alors tout langage L de $\mathcal{L}$ a un ensemble révélateur.*

Démonstration. Supposons qu'il existe un langage $L \in \mathcal{L}$ qui n'a pas d'ensemble révélateur. On construit inductivement les séquences $\langle a_0, a_1, \ldots\rangle$ et $\langle L_1, L_2, \ldots\rangle$:

- $i = 0$. Soit $a_0 \in L$.
- $i > 0$. L n'a pas d'ensemble révélateur et $\{a_0, \ldots, a_{i-1}\}$ est un ensemble fini, donc il existe un langage $L_i \in \mathcal{L}$ tel que $L_i \subset L$ et $\{a_0, \ldots, a_{i-1}\} \subseteq L_i$. Comme $L_i \subset L$, il existe un $a_i \in L$ tel que $a_i \notin L_i$.

L'existence des deux séquences $\langle a_0, a_1, \ldots\rangle$ et $\langle L_1, L_2, \ldots\rangle$ démontre l'élasticité infinie de $\mathcal{L}$. □

Cette proposition a pour conséquence l'apprenabilité *non effective* de toute classe de langages possédant l'élasticité finie, mais KAPUR montre dans [61] qu'une telle classe est de plus (effectivement) apprenable (voir la démonstration - et l'algorithme - dans la partie 2.4.1 ci-dessous). L'élasticité finie est une propriété structurelle des classes de langages (relativement) facile à démontrer, comparée au critère d'ANGLUIN d'énumération des ensembles révélateurs. C'est l'une des raisons pour lesquelles elle est souvent utilisée pour montrer l'apprenabilité. L'élasticité finie a de plus certaines propriétés remarquables (et utiles).

Une première propriété assez évidente de l'élasticité finie est que l'ensemble des classes de langages ayant cette propriété est fermée sous union. Ce n'est pas le cas pour l'ensemble des classes de langages apprenables, comme cela est démontré dans [48].

Proposition 2.8. *Si $\mathcal{L}_1$ et $\mathcal{L}_2$ sont deux classes de langages ayant l'élasticité finie, alors la classe $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$ a l'élasticité finie.*

Démonstration. Supposons que $\mathcal{L}$ a l'élasticité infinie. Alors il existe une séquence d'objets $\langle a_0, a_1, a_2, \ldots\rangle$ et une séquence de langages $\langle L_1, L_2, \ldots\rangle$ telles que $\{a_0, \ldots, a_{i-1}\} \subseteq L_i$ et $a_i \notin L_i$ pour tout $i > 0$. Pour tout i, L_i appartient à $\mathcal{L}_1$ ou $\mathcal{L}_2$, donc il existe $x \in \{1, 2\}$ et une séquence infinie d'indices $i_1, i_2, \ldots$ telle que pour tout $j > 0$ $L_{i_j} \in \mathcal{L}_x$. Par hypothèse on a $\{a_{i_1}, a_{i_2}, \ldots, a_{i_{j-1}}\} \subseteq \{a_0, a_1, \ldots, a_{i_j-1}\} \subseteq L_{i_j}$ et $a_{i_j} \notin L_{i_j}$, donc $\mathcal{L}_x$) a l'élasticité infinie (avec $x \in \{1, 2\}$. □

Nous présentons ci-dessous deux propriétés importantes de l'élasticité finie, découvertes par KAPUR et KANAZAWA.

2.4.1 Classes de langages uniformément apprenables

KAPUR étudie dans [61] la question de l'existence d'un algorithme d'apprentissage capable d'apprendre de la même manière différentes classes de langages. Un ensemble de classes de langages est

dit *uniformément apprenable* s'il existe un algorithme capable d'apprendre n'importe quel langage de n'importe quelle classe de cet ensemble.

KAPUR démontre d'abord que l'ensemble des classes apprenables n'est pas uniformément apprenable. Autrement dit, on ne peut concevoir une fonction d'apprentissage totalement universelle qui s'appliquerait à toute classe de langages apprenable. En revanche, il montre que certains ensembles de classes sont uniformément apprenables. C'est notamment le cas des classes de langages ayant l'élasticité finie.

Les *ensembles de test* définis par KAPUR sont comparables aux ensembles révélateurs d'ANGLUIN :

Définition 2.8 (Ensemble de test)**.** Soit $\mathcal{L}$ une classe de langages. Un ensemble fini d'objets D est un *ensemble de test* du langage $L \in \mathcal{L}$ si pour tout langage $L' \in \mathcal{L}$:

$$L \not\subseteq L' \Rightarrow D \cap (L \triangle L') \neq \emptyset.$$

Remarque : tout sur-ensemble d'un ensemble de test est aussi un ensemble de test.

Proposition 2.9. *Si D est un ensemble de test de L dans $\mathcal{L}$, alors $D \cap L$ est un ensemble révélateur de L dans $\mathcal{L}$.*

Démonstration. Soit D un ensemble de test de L. Si L' est un langage de $\mathcal{L}$ tel que $L' \subset L$, alors $L \not\subseteq L'$, donc $D \cap (L \triangle L') \neq \emptyset$. Ainsi il existe $x \in D$ tel que $x \in L \setminus L'$ ou $x \in L' \setminus L$. Or $L' \subset L$, donc $L' \setminus L = \emptyset$, d'où $x \in L \setminus L'$. Ainsi il existe $x \in D \cap L$ tel que $x \notin L'$, donc $D \cap L \not\subseteq L'$. □

La proposition 2.9 indique non seulement que si un ensemble de test existe alors un ensemble révélateur existe aussi, mais surtout le moyen de construire l'ensemble révélateur à partir de l'ensemble de test. En effet il suffit de calculer $D \cap L$, c'est-à-dire restreindre D, ensemble fini, aux éléments qui appartiennent à L.

La proposition suivante est l'équivalent de la proposition 2.6, relative à l'existence d'un point d'accumulation, pour les ensembles de test.

Proposition 2.10 (KAPUR)**.** *Soit $\mathcal{L}$ une classe de langages dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$. Un langage $L \in \mathcal{L}$ n'a pas d'ensemble de test si et seulement s'il existe une séquence infinie d'ensembles $\langle S_i \rangle_{i \in \mathbb{N}}$ telle que*

i. pour tout i, $Si \subseteq S_{i+1}$,

ii. $\bigcup_{i \in \mathbb{N}} S_i = \mathcal{U}$ et

iii. pour tout i, il existe $L' \in \mathcal{L}$ tel que $S_i \cap (L \triangle L') = \emptyset$ et $L \not\subseteq L'$.

Démonstration. $\Rightarrow$. Supposons que L n'a pas d'ensemble de test. L est donc infini, sinon L serait son propre ensemble de test (car $L \cap (L \triangle L') = L \cap ((L \setminus L') \cup (L' \setminus L)) = L \setminus L'$, et il est évident que si $L \not\subseteq L'$ alors $L \setminus L' \neq \emptyset$). Soit $\langle a_i \rangle_{i \in \mathbb{N}}$ une énumération de l'univers $\mathcal{U}$, et soit $S_i = \{a_0, \ldots, a_i\}$ pour tout i. On a bien $S_i \subseteq S_{i+1}$ (i) et l'union de tous les S_i est $\mathcal{U}$ (ii). Comme L n'a pas d'ensemble de test, pour tout D fini il existe $L' \in \mathcal{L}$ tel que $L \not\subseteq L'$ et $D \cap (L \triangle L') = \emptyset$. Or tous les S_i sont finis par construction, donc la dernière condition (iii) est remplie.

$\Leftarrow$. Supposons qu'il existe une séquence $\langle S_i \rangle_{i \in \mathbb{N}}$ vérifiant ces conditions. Quel que soit D un sous-ensemble fini de $\mathcal{U}$, il existe $n \in \mathbb{N}$ tel que $D \subseteq S_n$, car $S_i \subseteq S_{i+1}$ pour tout i, et l'union de tous les S_i est $\mathcal{U}$. Or d'après la condition (iii) il existe $L' \in \mathcal{L}$ tel que $S_n \cap (L \triangle L') = \emptyset$, donc D n'est pas un ensemble de test de L.

□

Proposition 2.11. *Soit $\mathcal{L}$ une classe de langages dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$. Si $\mathcal{L}$ a l'élasticité finie, alors tout langage $L \in \mathcal{L}$ a un ensemble de test dans $\mathcal{L}$.*

Démonstration. Supposons qu'il existe un langage L dans $\mathcal{L}$ qui n'a pas d'ensemble de test. D'après la proposition 2.10, il existe une séquence $\langle S_i \rangle_{i \in \mathbb{N}}$ telle que

i. pour tout i, $Si \subseteq S_{i+1}$,

ii. $\bigcup_{i \in \mathbb{N}} S_i = \mathcal{U}$ et

iii. pour tout i, il existe $L_i \in \mathcal{L}$ tel que $S_i \cap (L \triangle L_i) = \emptyset$ et $L \nsubseteq L_i$.

On montre d'abord qu'il existe une séquence d'indices $i_1, i_2, i_3 \ldots$ telle que pour tout j $L \cap S_{i_j} \subseteq L_{i_j}$ et $L \cap S_{i_{j+1}} \nsubseteq L_{i_j}$: soit i_1 le plus petit indice tel que S_{i_1} n'est pas vide. Pour tout j il existe L_{i_j} tel que $S_{i_j} \cap (L \triangle L_{i_j}) = \emptyset$, avec $L \nsubseteq L_{i_j}$. $L \cap S_{i_j} \subseteq L_{i_j}$, car $S_{i_j} \cap (L \triangle L_{i_j}) = \emptyset$. De plus $L \nsubseteq L_{i_j}$ implique qu'il existe $x \in L \setminus L_{i_j}$, donc comme x est dans $L \triangle L_{i_j}$ il faut que x n'appartienne pas à S_{i_j} pour que $S_{i_j} \cap (L \triangle L_{i_j}) = \emptyset$. Mais du fait que l'union de tous les S_i est $\mathcal{U}$, il existe nécessairement un $S_{i'}$, $i' > i_j$, tel que $x \in S_{i'}$. Soit $i_{j+1} = i'$: On a $x \in S_{i_{j+1}} \cap L$ et $x \notin L_{i_j}$, donc $L \cap S_{i_{j+1}} \nsubseteq L_{i_j}$.

À partir de cette séquence d'indices on peut montrer que $\mathcal{L}$ a l'élasticité infinie : pour tout j on a $L \cap S_{i_j} \subseteq L_{i_j}$ et $L \cap S_{i_{j+1}} \nsubseteq L_{i_j}$, donc il existe $a_j \in L \cap (S_{i_{j+1}} \setminus S_{i_j})$ tel que $a_j \notin L_{i_j}$. Pour tout $k < j$, $S_{i_{k+1}} \subseteq S_{i_j}$ donc $a_k \in L \cap (S_{i_{k+1}} \setminus S_{i_k}) \subseteq L \cap S_{i_j} \subseteq L_{i_j}$, d'où $\{a_0, \ldots, a_{j-1}\} \subseteq L_{i_j}$.

□

Proposition 2.12 (KAPUR)**.** *Soit $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ une famille indexée de langages récursifs dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$: $\mathcal{L} = \{ \mathcal{M}(G) \mid G \in \{G_0, G_1, G_2, \ldots\} \}$.*

Si tout langage $L \in \mathcal{L}$ a un ensemble de test dans $\mathcal{L}$, alors pour tout indice I il existe une procédure qui énumère un ensemble de test de $L_I = \mathcal{M}(G_I)$.

Démonstration. L'algorithme 2.3 renvoie les n premiers éléments d'une énumération d'un ensemble de test pour L_I (on note $L_i = \mathcal{M}(G_i)$ pour tout i).

Pour tout langage L_k tel que $L_I \nsubseteq L_k$, il existe un j suffisamment grand pour que $\{a_1, \ldots a_j\} \nsubseteq L_k$. La condition $D \cap (L_I \triangle L_k) = \emptyset$ est testée en vérifiant simplement que, pour tout $x \in D$, soit x appartient à la fois à L_I et L_k, soit x n'appartient à aucun des deux langages. Si cette condition est remplie, D n'est pas encore un ensemble de test pour L_I, c'est pourquoi on y ajoute les prochains éléments de l'énumération de $\mathcal{U}$. Par hypothèse un ensemble de test de L_I existe, donc il existe un i suffisamment grand pour que D contienne tous les éléments de cet ensemble de test. À chaque tour de boucle l'élément e_1 est ajouté arbitrairement à D, de manière à pouvoir compléter la séquence une fois que l'ensemble de test est complet.

□

EnumTest(G_I**,** n**)**
Soit $\langle e_1, e_2, \ldots \rangle$ une énumération de $\mathcal{U}$
$i \leftarrow 1$
$D \leftarrow \langle e_1 \rangle$
Soit $\langle a_1, a_2, \ldots \rangle$ une énumération de L_I
$j \leftarrow 1$
Tant que (**vrai**) **faire**
  Si (D contient plus de n éléments) **Alors**
    Renvoyer les n premiers éléments de la séquence D
  Fin Si
  Pour k **de** 1 **à** j **faire**
    Si ($\{a_1, \ldots, a_j\} \not\subseteq L_k$) **Alors**
      Tant que ($D \cap (L_I \triangle L_k) = \emptyset$) **faire**
        $i \leftarrow i + 1$
        $D \leftarrow D \bullet \langle e_i \rangle$
      Fin Tant Que
    Fin Si
  Fin Pour
  $j \leftarrow j + 1$
  $D \leftarrow D \bullet \langle e_1 \rangle$
Fin Tant Que

ALG. 2.3 – Énumération d'un ensemble de test

Comme on peut facilement passer d'un ensemble de test D à un ensemble révélateur (d'après la proposition 2.9), les propositions 2.11 et 2.12 suffisent à construire un algorithme d'apprentissage pour toute classe de langages ayant l'élasticité finie. KAPUR va plus loin en proposant une nouvelle condition équivalente à l'apprenabilité (effective), basée sur les ensembles de tests.

Proposition 2.13 (KAPUR)**.** *Soit $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ une famille indexée de langages récursifs dans le système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$: $\mathcal{L} = \{ \mathcal{M}(G) \mid G \in \{G_0, G_1, G_2, \ldots\} \}$.*

Il existe une fonction ϕ qui apprend $\mathcal{L}$ si et seulement s'il existe un algorithme qui, pour tout langage $L_I = \mathcal{M}(G_I) \in \mathcal{L}$, énumère récursivement une famille indexée $\mathcal{L}_I \subseteq \mathcal{L}$ telle que

- *Tout langage $L' \in \mathcal{L}$ tel que $L' \subseteq L_I$ appartient à $\mathcal{L}_I$, et*
- *L_I a un ensemble de test dans $\mathcal{L}_I$.*

Démonstration. $\Rightarrow$. Supposons qu'une fonction d'apprentissage existe. D'après la proposition 2.2, il existe un algorithme $EnumRevel$ qui énumère un ensemble révélateur de L_i pour tout $L_i \in \mathcal{L}$. L'algorithme 2.4 énumère la famille indexée $\mathcal{L}_I$.

EnumSousFamille(G_I,n)
$F \leftarrow \{G_I\}$
$i \leftarrow 1$
Tant que (**vrai**) **faire**
  Si (F contient plus de n éléments) **Alors**
    Renvoyer les n premiers éléments de F
  Fin Si
  $j \leftarrow 1$
  Pour j **de** 1 **à** i **faire**
    Si ($EnumRevel(I, i) \notin \mathcal{M}(G_j)$) **Alors**
      $F \leftarrow F \bullet \langle G_j \rangle$
    Sinon
      $F \leftarrow F \bullet \langle G_I \rangle$
    Fin Si
  Fin Pour
  $i \leftarrow i + 1$
Fin Tant Que

ALG. 2.4 – Énumération d'une sous-famille à partir de l'ensemble révélateur

Soit $D = \{ x = EnumRevel(I, i) \mid i \in \mathbb{N} \}$ est un ensemble révélateur de L_I, donc pour i suffisamment grand tout langage $L_j \subset L_I$ est tel que $EnumRevel(I, i) \notin \mathcal{M}(G_j)$. Par conséquent tout $L' \in \mathcal{L}$ tel que $L' \subseteq L_I$ appartient à $\mathcal{L}_I$. De plus D constitue un ensemble de test de L_I dans $\mathcal{L}_I$, car $D \subseteq L_I$ mais pour tout $L_j \in \mathcal{L}_I$ $D \nsubseteq L_j$, d'où $D \cap (L_I \triangle L_j) \neq \emptyset$.

$\Leftarrow$. Supposons qu'il existe un algorithme qui, pour tout langage $L_I \in \mathcal{L}$, énumère récursivement une famille indexée $\mathcal{L}_I$ telle que tout $L' \in \mathcal{L}$ tel que $L' \subseteq L_I$ appartient à $\mathcal{L}_I$, et L_I a un ensemble

de test dans $\mathcal{L}_I$. D'après la proposition 2.12, il existe un algorithme $EnumTest(I, n)$ qui énumère un tel ensemble D. En utilisant le fait que $D \cap L_I$ est un ensemble révélateur de L_I (proposition 2.9), on construit l'algorithme 2.5 qui énumère cet ensemble.

EnumRevelateur(G_I,n)
Soit a_i le $i^{\text{ème}}$ élément de $EnumTest(I, i)$ pour tout i.
Soit $x = a_i$, avec i le plus petit entier tel que $a_i \in L_I$.
$D \leftarrow \langle\rangle$
Pour i **de** 1 **à** n **faire**
  Si $(a_i \in L_I)$ **Alors**
    $D \leftarrow D \bullet \langle a_i \rangle$
  Sinon
    $D \leftarrow D \bullet \langle x \rangle$
  Fin Si
Fin Pour
Renvoyer D

ALG. 2.5 – Énumération d'un ensemble révélateur à partir d'un ensemble de test

Tout $L' \in \mathcal{L}$ tel que $L' \subseteq L_I$ appartient à $\mathcal{L}_I$, donc l'ensemble révélateur de L_I dans $\mathcal{L}_I$ ainsi énuméré est aussi un ensemble révélateur de L_I dans $\mathcal{L}$. D'après la proposition 2.2, une fonction d'apprentissage de $\mathcal{L}$ existe.

□

Cette condition nécessaire et suffisante pour l'apprenabilité proposée par KAPUR est une extension précieuse du théorème d'ANGLUIN. Rappelons que la condition d'ANGLUIN repose sur l'existence d'un ensemble révélateur pour chaque langage de la classe, mais qu'il faut aussi montrer l'existence d'un algorithme qui énumère les ensembles révélateurs. Or même lorsque l'existence d'un tel algorithme est démontrée, il n'y a pas de stratégie générale de construction de cet algorithme. Le théorème 2.13 montre non seulement que la seule existence des ensemble de tests suffit pour l'apprenabilité, mais donne aussi l'algorithme qui apprend uniformément la classe de langages. L'énumérabilité d'une sous-famille sert à rendre la condition à la fois nécessaire et suffisante, ainsi le théorème permet d'inclure toutes les classes de langages apprenables.

D'après la proposition 2.11, toute classe de langages ayant l'élasticité finie admet un ensemble de test pour chacun de ses langages. Ainsi l'élasticité finie correspond au cas particulier du théorème de KAPUR dans lequel la classe $\mathcal{L}_I$ à énumérer peut simplement être la classe $\mathcal{L}$ elle-même (il s'agit d'une famille indexée, qui peut par définition être énumérée par un algorithme).

Corollaire 2.3. *Si la classe de langages $\mathcal{L}$ a l'élasticité finie, alors $\mathcal{L}$ est apprenable.*

2.4.2 Relation finiment valuée

L'élasticité finie présente aussi des propriétés utiles pour démontrer l'apprenabilité de classes de langages non triviales. La première de ces propriétés est démontrée par WRIGHT dans [109] : si deux classes de langages $\mathcal{L}$ et $\mathcal{L}'$ ont toutes les deux l'élasticité finie, alors la classe $\{ L \cup L' \,|\, L \in \mathcal{L} \text{ et } L' \in \mathcal{L}' \}$ a aussi l'élasticité finie. KANAZAWA donne dans [59] une version beaucoup plus riche de ce théorème. Sa généralisation repose sur l'utilisation d'une *relation finiment valuée*.

Définition 2.9 (Relation finiment valuée)**.** Soient A et B deux ensembles. Une relation binaire R sur $A \times B$ est *finiment valuée* si et seulement si pour tout $a \in A$ il n'existe qu'un nombre fini de $b \in B$ tels que $a\ R\ b$.

Remarque : Cette définition n'est pas symétrique : si R est une relation finiment valuée sur $A \times B$, une relation R' sur $B \times A$ telle que $b\ R'\ a$ si et seulement si $a\ R\ b$ n'est pas nécessairement finiment valuée.

Définition 2.10 (Image inverse d'un langage)**.** Si L est un langage sur $\mathcal{U}$ et R une relation sur $\mathcal{U}' \times \mathcal{U}$, l'*image inverse* de L par rapport à R est le langage $R^{-1}[L] = \{\ a \in \mathcal{U}' \mid \exists b \in L \text{ tel que } a\ R\ b\ \}$.

Proposition 2.14 (KANAZAWA)**.** *Soient $\mathcal{U}$ et $\mathcal{U}'$ deux ensembles d'objets, et $\mathcal{L}$ une classe de langages définie sur $\mathcal{U}$ qui a l'élasticité finie.*

S'il existe une relation $R \subseteq \mathcal{U}' \times \mathcal{U}$ finiment valuée, alors la classe de langages $\mathcal{L}' = \{\ R^{-1}[L] \,|\, L \in \mathcal{L}\ \}$ a aussi l'élasticité finie.

Démonstration. Supposons que $\mathcal{L}' = \{\ R^{-1}[L] \mid L \in \mathcal{L}\ \}$ a l'élasticité infinie : il existe une séquence d'objets de $\mathcal{U}'$ $\langle a_i \rangle_{i \in \mathbb{N}}$ et une séquence de langages de $\mathcal{L}'$ $\langle L'_i \rangle_{i \in \mathbb{N}^*}$ tels que pour tout i $\{a_0, \ldots, a_{i-1}\} \subseteq L'_i$ et $a_i \notin L'_i$. Pour tout $i > 0$ il existe un langage $L_i \in \mathcal{L}$ tel que $L'_i = R^{-1}[L_i]$.

Pour tout $i \geq 0$ on définit l'ensemble de séquences

$$E_i = \{\ \langle b_0, \ldots, b_i \rangle \mid \forall j, 0 \leq j \leq i, a_j\ R\ b_j \text{ et } \exists n \text{ tel que } \{b_0, \ldots, b_i\} \subseteq L_n\ \}.$$

Cet ensemble n'est pas vide, puisque pour tout $k > i$ $\{a_0, \ldots, a_i\} \subseteq L'_k = \{\ a \in \mathcal{U}' \mid \exists b \in L_k \text{ tel que } a\ R\ b\ \}$. De plus E_i et E_j sont disjoints si $i \neq j$ (car la longueur des séquences dans E_i est différente de celle dans E_j), donc l'ensemble

$$E = \bigcup_{i \in \mathbb{N}} E_i$$

est infini. E forme un arbre, en prenant le nœud $\langle b_0, \ldots, b_i \rangle \in E_i$ comme père du nœud $\langle b_0, \ldots, b_i, b_{i+1} \rangle \in E_{i+1}$, pour toute séquence $\langle b_0, \ldots, b_i, b_{i+1} \rangle \in E_{i+1}$ (la racine est la séquence vide). Comme R est une relation finiment valuée, il ne peut y avoir qu'un nombre fini de b_i tels que $a_i\ R\ b_i$, donc il n'y a qu'un nombre fini de b_{i+1} dans E_{i+1} qui peuvent être ajoutés à une séquence $\langle b_0, \ldots, b_i \rangle$ fixée. Ainsi chaque nœud dans l'arbre E ne peut avoir qu'un nombre fini de fils. Mais comme E est infini, il y a nécessairement une branche infinie dans E (lemme de König). Soit $\langle b_0 \rangle, \langle b_0, b_1 \rangle, \langle b_0, b_1, b_2 \rangle, \ldots$ les nœuds de cette branche infinie. $b_i \notin L_i$, car $a_i \notin L'_i$ implique qu'il n'existe aucun b dans L_i tel que $a_i\ R\ b$.

Soit $f(n)$ la fonction définie par $\{b_0, \ldots, b_n\} \subseteq L_{f(n)}$ et pour tout $j < f(n)$, $\{b_0, \ldots, b_n\} \nsubseteq L_j$ (autrement dit, $f(n)$ est le plus petit entier tel que $\{b_0, \ldots, b_n\} \subseteq L_{f(n)}$). On observe que $n < f(n)$, car $b_n \notin L_n$. Pour tout $n \in \mathbb{N}$, on définit inductivement la fonction g par

$$\begin{cases} g(0) = 0 \\ g(n) = f(g(n-1)) \ , \ \forall n > 0 \end{cases}$$

$g(n) = f(g(n-1)) > g(n-1)$, donc g est strictement croissante, d'où $\{b_{g(0)}, b_{g(1)}, \ldots, b_{g(n-1)}\} \subseteq \{b_0, b_1, \ldots, b_{g(n-1)}\}$. Or $\{b_0, b_1, \ldots, b_{g(n-1)}\} \subseteq L_{f(g(n-1))} = L_{g(n)}$, on obtient par conséquent les deux séquences infinies $\langle b_{g(n)} \rangle_{n \in \mathbb{N}}$ et $\langle L_{g(n)} \rangle_{n \in \mathbb{N}^*}$ telles que pour tout $n > 0$ $\{b_{g(0)}, \ldots, b_{g(n-1)}\} \subseteq L_{g(n)}$ et $b_{g(n)} \notin L_{g(n)}$ (car $b_i \notin L_i$ pour tout i). □

Ce théorème s'avère très utile pour démontrer l'élasticité finie d'une classe dans le cas où il existe une classe "voisine" pour laquelle l'élasticité finie est déjà prouvée. Il est ainsi possible d'étendre l'élasticité finie à une classe plus complexe, dès lors qu'une relation finiment valuée peut être établie entre les univers de définition des deux classes telle que la classe "cible" est entièrement contenue dans l'image inverse de la classe "source" par cette relation. Cette méthode est comparable à la réduction d'un problème à un autre, utilisée par exemple pour montrer la NP-complétude du problème.

Le théorème de WRIGHT est alors un cas particulier assez simple de la proposition 2.14, comme le montre KANAZAWA :

Exemple 2.3 ($\mathcal{L}_1$ et $\mathcal{L}_2$ ont l'élasticité finie $\Rightarrow$ $\{ L_1 \cup L_2 \mid L_1 \in \mathcal{L}_1$ et $L_2 \in \mathcal{L}_2 \}$ a l'élasticité finie)**.** Soit $\mathcal{L}_1$ et $\mathcal{L}_2$ deux classes de langages sur $\mathcal{U}_1$ et $\mathcal{U}_2$ (respectivement) ayant l'élastitictéfinie. Soit x_1 et x_2 deux symboles qui n'apparaissent ni dans $\mathcal{U}_1$ ni dans $\mathcal{U}_2$. Pour tout couple de langages, on définit l'opération $\uplus$ par $L_1 \uplus L_2 = \{ bx_1 \mid b \in L_1 \} \cup \{ bx_2 \mid b \in L_2 \}$.

On montre par contraposée que la classe $\{ L_1 \uplus L_2 \mid L_1 \in \mathcal{L}_1$ et $L_2 \in \mathcal{L}_2 \}$ a l'élasticité finie. Supposons que ce ne soit pas le cas. Alors il existe une séquence infinie d'objets $\langle a_0, a_1, \ldots \rangle$ et deux séquences infinies de langages $\langle L_1^1, L_2^1, \ldots \rangle$ et $\langle L_1^2, L_2^2, \ldots \rangle$ telles que pour tout $i > 0$ $\{a_0, \ldots, a_{i-1}\} \subseteq L_i^1 \uplus L_i^2$ et $a_i \notin L_i^1 \uplus L_i^2$. Tout a_i est de la forme bx_1 ou bx_2, donc il existe $k \in \{1, 2\}$ tel qu'il y a une séquence infinie $\langle a_{i_0}, a_{i_1}, a_{i_2}, \ldots \rangle$ avec a_{i_j} est de la forme $b_{i_j} x_k$ pour tout j. Pour tout $j > 0$, $a_{i_j} = b_{i_j} x_k \notin L_{i_j}^1 \uplus L_{i_j}^2$ donc $b_{i_j} \notin L_{i_j}^k$. De plus comme $\{a_{i_0}, a_{i_1}, \ldots, a_{i_{j-1}}\} \subseteq \{a_0, a_1 \ldots, a_{i_j - 1}\} \subseteq L_{i_j}^1 \uplus L_{i_j}^2$ on a $\{b_{i_0}, b_{i_1}, \ldots, b_{i_{j-1}}\} \subseteq L_{i_j}^k$. On obtient donc deux séquences qui montrent l'élasticité infinie de $\mathcal{L}_k$.

Soit R la relation finiment valuée définie de telle sorte que $b \ R \ a$ si et seulement si $a = bx_1$ ou $a = bx_2$. Soit $\mathcal{L} = \{ R^{-1}[L_1 \uplus L_2] \mid L_1 \in \mathcal{L}_1$ et $L_2 \in \mathcal{L}_2 \}$. On voit que $\mathcal{L} = \{ L_1 \cup L_2 \mid L_1 \in \mathcal{L}_1$ et $L_2 \in \mathcal{L}_2 \}$. D'après le théorème 2.14, $\mathcal{L}$ a l'élasticité finie.

2.5 La densité finie bornée (SHINOHARA)

Les résultats présentés jusqu'ici sont des critères très généraux d'apprenabilité, fondés (quasiment) uniquement sur des propriétés des langages. Bien sûr, ces résultats sont applicables à des systèmes de grammaires concrets. Néanmoins il s'agit de résultats relativement abstraits, qui ne sont liés à aucune

formalisme particulier. SHINOHARA propose dans [95], [94], [10], [96] de descendre au niveau des grammaires pour étudier l'apprenabilité, tout en conservant un assez haut niveau de généralité.

Dans [95], SHINOHARA définit la notion de *cadre de définition de concepts* (*concept defining franmework*) dans les termes suivants : il s'agit d'un triplet $\langle U, E, M\rangle$, dans lequel U est l'univers (ensemble des objets), E est l'ensemble des expressions et M une correspondance sémantique. Un concept est un sous-ensemble $R \subseteq U$, et un système formel est un ensemble fini $\Gamma \subseteq E$. Une correspondance sémantique est une relation M associant des concepts à des systèmes formels : lorsque $M(\Gamma) = R$ on dit que le système formel Γ définit le concept R.

Pour reprendre notre terminologie, un cadre de définition de concepts est un système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M}\rangle$ dans lequel l'ensemble des grammaires $\mathcal{G}$ n'est constitué que d'ensembles finis d'expressions prises dans l'ensemble E. Afin de conserver une homogénéité des notations, nous présentons ci-dessous les résultats de SHINOHARA avec nos propres notations. Il faut toutefois garder à l'esprit que les grammaires considérées ne sont que des ensembles d'expressions. Ceci autorise entre autres l'utilisation de la relation $\subseteq$ ou de $|G|$ sur ce type de grammaires.

Définition 2.11 (Monotonie)**.** La correspondance sémantique $\mathcal{M}$ est monotone si $G \subseteq G'$ implique $\mathcal{M}(G) \subseteq \mathcal{M}(G')$.

Définition 2.12 (Grammaire réduite)**.** Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M}\rangle$ un système de grammaires.

Une grammaire $G \in \mathcal{G}$ est réduite par rapport à un ensemble $D \subseteq \mathcal{U}$ si $D \subseteq \mathcal{M}(G)$ mais $D \nsubseteq \mathcal{M}(G')$ pour toute grammaire $G' \in \mathcal{G}$ telle que $G' \subset G$.

Définition 2.13 (Densité finie bornée)**.** Un système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M}\rangle$ a la densité finie bornée[17] si $\mathcal{M}$ est monotone et l'ensemble

$$\{\ \mathcal{M}(G) \mid G \text{ est réduite par rapport à } D \text{ et } |G| \leq n\ \}$$

est fini pour tout ensemble fini $D \subseteq \mathcal{U}$ et tout $n \geq 0$.

Théorème 2.3. *Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M}\rangle$ un système de grammaire et $\mathcal{L}_n = \{\ \mathcal{M}(G) \in \mathcal{G} \mid |G| \leq n\ \}$ pour tout $n \in \mathbb{N}$.*

Si $\langle \mathcal{U}, \mathcal{G}, \mathcal{M}\rangle$ a la densité finie bornée, alors la classe $\mathcal{L}_n$ a l'élasticité finie pour tout $n \in \mathbb{N}$.

La démonstration de ce théorème sera donnée dans le chapitre 5 (théorème 5.1) (dans une version généralisée). Bien entendu, le fait que toute classe $\mathcal{L}_n$ a l'élasticité finie implique que celle-ci est apprenable (selon le corollaire 2.3).

[17] Nous reprenons dans ce document la traduction française *densité finie bornée* pour *bounded finite thickness* proposée par BÉCHET et FORET dans [16].

Malgré son apparence relativement simple, ce résultat de SHINOHARA s'avère un outil très puissant pour démontrer l'apprenabilité de classes de langages qui n'ont rien de trivial. SHINOHARA en propose plusieurs applications dans [95] : sur les systèmes formels élémentaires, les grammaires contextuelles et certaines formes de programmes Prolog. Les systèmes formels élémentaires et les programmes Prolog ne présentent pas ou peu d'intérêt pour la représentation des langues naturelles, nous illustrerons donc ci-dessous le théorème 2.3 à l'aide du résultat très important de SHINOHARA sur les langages contextuels (voir partie 1.3.1).

Proposition 2.15. *Soit $D \subseteq \Sigma^*$ un ensemble fini de phrases sur un vocabulaire Σ.*

Si G est une grammaire contextuelle réduite par rapport à D, alors pour toute règle $A_1 \dots A_n \mapsto B_1 \dots B_m$ dans G on a $n \leq m \leq \max(\{\ |x| \mid x \in D\ \})$.

Démonstration. Soit G une grammaire réduite par rapport à D et $x \in D$.

Soit $S \rightarrow_G a_1^1 a_2^1 \dots a_{n_1}^1 \rightarrow_G a_1^2 a_2^2 \dots a_{n_2}^2 \rightarrow_G \dots \rightarrow_G a_1^p a_2^p \dots a_{n_p}^p$ une dérivation complète de x pour la grammaire G, avec $x = a_1^p a_2^p \dots a_{n_p}^p$ et $a_i^p \in \Sigma$ pour tout i. Par définition, $a_1^j a_2^j \dots a_{n_j}^j \rightarrow_G a_1^{j+1} a_2^{j+1} \dots a_{n_{j+1}}^{j+1}$ si et seulement s'il existe une règle $A_1 \dots A_n \mapsto B_1 \dots B_m$ dans G et un entier k ($k \geq 0$, $k \leq n_j - n$ et $k \leq n_{j+1} - m$) tels que

- $\langle a_{k+1}^j, a_{k+2}^j, \dots, a_{k+n}^j \rangle = \langle A_1, A_2, \dots, A_n \rangle$,
- $\langle a_{k+1}^{j+1}, a_{k+2}^{j+1}, \dots, a_{k+m}^{j+1} \rangle = \langle B_1, B_2, \dots, B_m \rangle$,
- et $\langle a_1^j, \dots, a_k^j \rangle = \langle a_1^{j+1}, \dots, a_k^{j+1} \rangle$ et $\langle a_{k+n+1}^j, \dots, a_{n_j}^j \rangle = \langle a_{k+m+1}^{j+1}, \dots, a_{n_{j+1}}^{j+1} \rangle$.

Ainsi $n_{j+1} = k + m + (n_j - k - n) = n_j + m - n$. Or G est contextuelle donc par définition $n \leq m$, ce qui a pour conséquence que $n_{j+1} \geq n_j$. Ceci étant valable pour tout $1 \leq j \leq p$, on obtient que $n_j \leq n_p = |x|$ pour tout j. Il est clair que toute règle $A_1 \dots A_n \mapsto B_1 \dots B_m$ utilisée dans la dérivation vérifie $n \leq m \leq n_j$ pour un certain j, donc $n \leq m \leq |x|$.

Par conséquent, toute règle $A_1 \dots A_n \mapsto B_1 \dots B_m$ utilisée dans une dérivation d'un élément x de D vérifie $n \leq m \leq \max(\{\ |x| \mid x \in D\ \})$. Supposons que G contient une règle R qui n'est utilisée dans aucune de ces dérivations : dans ce cas G n'est pas réduite par rapport à D, puisque la grammaire $G' = G \setminus \{R\}$ vérifie à la fois $D \subseteq L(G')$ et $G' \subset G$. Donc G ne contient bien que des règles vérifiant $n \leq m \leq \max(\{\ |x| \mid x \in D\ \})$. □

Proposition 2.16. *Soit $\mathcal{G}$ l'ensemble des grammaires contextuelles.*

Le système de grammaires $\langle \Sigma^*, \mathcal{G}, L \rangle$ a la densité finie bornée.

Démonstration. Clairement la relation L associant à toute grammaire son langage est monotone. Soit $D \subseteq \Sigma^*$, $k \in \mathbb{N}$ et $G \in \mathcal{G}$ une grammaire contextuelle réduite par rapport à D telle que $|G| \leq k$. Selon la proposition 2.15, toute règle $A_1 \dots A_n \mapsto B_1 \dots B_m$ dans G vérifie $n \leq m \leq \max(\{\ |x| \mid x \in D\ \})$, donc la longueur d'une telle règle est bornée par $l = 2 \times \max(\{\ |x| \mid x \in D\ \})$. De plus G ne contient pas plus de n règles, donc la taille globale de G est bornée par $n \times l$.

Ainsi, si l'on ne tient pas compte des variantes par renommage des symboles non terminaux, le nombre de grammaires différentes de taille inférieure ou égale à $n \times l$ est borné. Or toute variante G' par

renommage des symboles non terminaux d'une grammaire G vérifie $L(G) = L(G')$, donc cela implique que l'ensemble

$$\{\ L(G) \mid G \text{ est réduite par rapport à } D \text{ et } |G| \leq k\ \}$$

est fini, pour tout $D \subseteq \Sigma^*$ et tout $k \geq 0$. □

Corollaire 2.4. *Pour tout $k \in \mathbb{N}$, la classe*

$$\{\ L(G) \mid G \textit{ est une grammaire contextuelle d'au plus } k \textit{ règles}\ \}$$

a l'élasticité finie (et est donc apprenable).

Démonstration. Simple application du théorème 2.3 (et du corollaire 2.3). □

2.6 Conclusion

2.6.1 Application(s) aux langues naturelles ?

Le modèle de GOLD est un cadre formel possible du processus de l'apprentissage. Mais est-il pertinent de l'utiliser pour représenter l'acquisition de langues naturelles ? Tout d'abord, ce modèle théorique semble répondre relativement bien à ce qu'on en attend intuitivement : l'apprenant est face à une séquence infinie d'exemples du langage, et fait évoluer sa conception de ce langage en fonction des exemples vus. L'apprenant n'est jamais informé du fait qu'il a atteint le langage cible, mais le modèle garantit (dans le cas d'une classe apprenable) qu'il est possible de l'atteindre avec un nombre fini d'exemples. De plus, certaines caractéristiques du modèle sont en accord avec les observations linguistiques sur l'acquistion du langage par l'enfant : l'enfant apprend sa langue maternelle en utilisant uniquement les exemples positifs dont il dispose (les phrases du langage prononcées par son entourage) [88], [65]. Enfin, le résultat de SHINOHARA [95] sur l'apprenabilité des langages définis par des grammaires contextuelles d'au plus k règles constitue sans doute l'un des points positifs les plus importants du modèle : si l'on suppose qu'il est possible de représenter tout langage naturel sous forme d'une grammaire contextuelle finie, alors selon ce résultat la classe des langues naturelles est apprenable dans le modèle de GOLD, puisqu'il existe un nombre k suffisamment grand pour représenter l'ensemble des langues naturelles sous forme de grammaires contextuelles.

Même si la définition du modèle décrit l'apprenant sous la forme d'une fonction mathématique, il n'y a aucune contradiction à considérer ce modèle du point de vue de l'apprentissage humain. Cet aspect est d'ailleurs l'une des motivations de GOLD, qui étudie dans [50] les implications de ses résultats par rapport aux études sur l'acquisition du langage par l'enfant. Son modèle est assez largement connu et utilisé (ou critiqué) en sciences cognitives, comme en témoigne par exemple l'article de JOHNSON [57], dans lequel différentes interprétations du théorème 2.1 sont discutées.

Néanmoins, il ne faut pas prêter à ce modèle plus de qualités qu'il n'en a. Par exemple, l'infinité de la séquence d'exemples positifs ne doit pas être confondue avec l'aspect dynamique des langues : dans le modèle de GOLD, le postulat de base est que le langage existe dans la classe étudiée, et est par conséquent fixé, même si on ne sait pas encore le représenter, dès le premier exemple. En particulier, il est possible qu'un mot n'apparaisse que "très tard" dans la séquence de phrases, mais l'ensemble du vocabulaire est fixé dès le départ (sans quoi la notion de convergence n'aurait aucun sens). Un autre aspect important pour les langues naturelles sur lequel le modèle reste très abstrait est la représentation des langages. On peut en effet noter que les éléments de base dans cette définition de l'apprenabilité sont les *objets* du langage, compris comme étant de simples éléments d'un ensemble. Or dans le cas d'un langage naturel, les objets sont des phrases (éventuellement enrichies d'informations sur leur structure), ce qui signifie que le modèle n'impose aucune contrainte sur (par exemple) la décomposition en mots ou la sémantique. Dans un sens il s'agit bien sûr d'un avantage, puisque cela laisse toute liberté sur le choix du formalisme utilisé, mais cela montre aussi que le modèle n'est pas exclusivement conçu pour les langues naturelles.

On peut distinguer deux directions relativement distinctes dans l'application du modèle de GOLD aux langues naturelles. La première direction travaille sur l'hypothèse que ce modèle est une représentation théorique fiable du processus d'acquisition du langage par l'enfant. Dans ce cadre, l'objectif consiste principalement à valider ou invalider des hypothèses sur le mode d'apprentissage de l'enfant, notamment en simulant cette acquisition par la machine. Par exemple, DUDAU-SOFRONIE et TELLIER [43] [42] adaptent les résultats de Kanazawa sur les grammaires catégorielles de manière à prendre en compte des informations d'ordre sémantique dans les exemples, conformément à ce dont dispose un enfant dans son environnement. La seconde direction, à laquelle se rattache notre approche, vise à faire mettre en pratique l'apprentissage par la machine, dans l'objectif d'en tirer des informations (idéalement, une grammaire précise et exhaustive) potentiellement utilisables à d'autres tâches. Ce passage de la théorie à la pratique pose les problèmes complexes de l'existence / l'obtention de données correctes comme séquence d'exemples (avec pour corollaire la question de la nature des données - voir la partie 7.2.1) et évidemment de l'efficacité des algorithmes.

Dans les deux cas, il faut que l'apprenabilité de la classe de langages considérée soit démontrée avant d'envisager la construction d'un algorithme d'apprentissage : vouloir établir un tel algorithme sans passer par cette première étape serait en quelque sorte comme tenter de réaliser un algorithme pour résoudre un problème dont on ignore s'il est décidable. Cette recherche de l'apprenabilité est fortement liée au formalisme grammatical utilisé, puisqu'il est souvent nécessaire de restreindre la classe de langages en fonction de critères propres à ce formalisme (comme dans le cas des grammaires contextuelles dont le nombre de règles est borné). Mais si la preuve d'apprenabilité formelle induit souvent un algorithme convergent, elle ne garantit pas en revanche l'efficacité de celui-ci[18]. La mise à l'échelle est donc certainement le principal obstacle à l'utilisation du modèle de GOLD en traitement automatique des langues, puisque la plupart des algorithmes "intéressants" dans ce cadre sont de complexité exponentielle.

[18] Dans le cas de l'apprenabilité non effective, celle-ci n'est même pas une condition suffisante à la simple existence d'un algorithme d'apprentissage.

2.6.2 De la théorie à la pratique ?

En 1983, ANGLUIN concluait son état de l'art du domaine de l'inférence inductive [8] par les remarques suivantes :

> *"Le problème ouvert le plus important n'est sans doute pas une quelconque question technique spécifique, mais le fossé entre les résultats abstraits et concrets. Il serait malheureux que les résutats théoriques se soient multipliés en vain, tandis que les résultats appliqués n'aient eu que peu ou aucun impact au-delà de leur domaine restreint. [...]*
> *En dépit de ces problèmes, nous pensons que les questions de base en inférence inductive sont suffisamment intéressantes et importantes pour que de bons travaux continuent de les éclaircir, et les générations futures riront bien de notre ignorance actuelle."* *[8]*

Plus de vingt ans plus tard, il est vrai qu'un certain nombre de travaux brillants ont largement contribué à une meilleure compréhension des mécanismes mis en jeu dans le modèle de GOLD, avec des conséquences importantes sur les possibilités théoriques d'apprentissage dans ce cadre. Mais le fossé entre ces résultats théoriques et les aspects applicatifs de l'apprentissage est loin d'avoir été comblé.

Dans un sens, le modèle de GOLD a fait ses preuves en tant que système formel représentant le processus d'apprentissage. En effet, contrairement aux premières impressions, les résultats négatifs obtenus dès le départ montrent seulement que le modèle n'est pas trivial, au sens où il ne permet pas d'apprendre "n'importe quoi". Les résultats positifs obtenus par la suite démontrent que le modèle est finalement relativement équilibré, puisqu'il permet d'apprendre (de manière non évidente) des classes de langages très riches. Le fait que la frontière entre langages apprenables et non apprenables ne corresponde pas à la hiérarchie traditionnelle définie par CHOMSKY importe peu. Du point de vue cognitif, le modèle ne présente pas de contradiction avec l'acquisition humaine du langage. KAPUR indique que *"Plutôt que de considérer le paradigme de Gold comme totalement inapproprié, [nous soutenons] l'idée que le modèle comporte une flexibilité adéquate pour représenter certains aspects de l'acquisition de langues naturelles."* [61].

En revanche, le modèle de GOLD demeure avant tout un modèle théorique, non appliqué, et on pourrait même douter qu'il soit applicable en général. L'existence d'algorithmes dits *universels* d'apprentissage dans ce modèle ne peut que servir de preuve de la décidabilité du problème, car ces algorithmes d'énumération sont de toute évidence totalement impraticables[19]. Il ne faut cependant pas conclure trop vite à l'inadéquation du modèle, puisque le problème de l'inférence grammaticale en lui-même, quelle que soit la définition qu'on en prend, est extrêmement complexe. Nous verrons dans les prochains chapitres les conséquences sur des formalismes grammaticaux précis de ce fossé entre apprenabilité théorique et pratique, et les problèmes que cela pose pour l'apprentissage appliqué aux langues naturelles.

[19] C'est la raison pour laquelle nous avons adopté un point de vue plutôt algorithmique tout au long de ce chapitre. Nous entendons souligner ainsi combien la notion d'apprenabilité est éloignée des applications pratiques : la plupart des résultats d'apprenabilité (par exemple ceux basés sur l'élasticité finie) assurent seulement l'existence d'une méthode d'apprentissage basée sur l'énumération d'ensembles révélateurs (algorithme 2.2, p. 54).

CHAPITRE 3

Apprentissage de grammaires catégorielles

JUSQU'À LA FIN DES ANNÉES 1990, IL N'Y A QUASIMENT AUCUN LIEN ENTRE LE MODÈLE D'APPRENTISSAGE DÉFINI PAR GOLD ET LES FORMALISMES GRAMMATICAUX ADAPTÉS AUX LANGUES NATURELLES. KANAZAWA RÉALISE CETTE JONCTION DANS [60], EN DÉMONTRANT TOUTE UNE SÉRIE DE RÉSULTATS CONCERNANT LES GRAMMAIRES CATÉGORIELLES CLASSIQUES. DU POINT DE VUE DE LA REPRÉSENTATION DES LANGUES NATURELLES IL S'AGIT D'UNE AVANCÉE MODESTE, PUISQUE LES GRAMMAIRES AB SONT TRÈS LIMITÉES. MAIS LES TRAVAUX DE KANAZAWA SONT UN PREMIER PAS CONCRET POUR LE MODÈLE DE GOLD EN DIRECTION DES GRAMMAIRES CATÉGORIELLES, POTENTIELLEMENT RICHES EN TERMES DE REPRÉSENTATION DES LANGUES. DE PLUS, KANAZAWA DÉPLOIE CERTAINES TECHNIQUES NOUVELLES D'EXPLORATION DE L'APPRENABILITÉ DES LANGAGES, AUSSI BIEN AU NIVEAU THÉORIQUE QU'ALGORITHMIQUE. SON TRAVAIL OUVRE AINSI DE NOMBREUSES PERSPECTIVES D'APPLICATION DU MODÈLE DE GOLD AUX LANGUES NATURELLES.

3.1 Introduction

Après des débuts assez peu prometteurs, le modèle proposé par GOLD dans [50] s'est finalement vu adopté comme représentation riche et pertinente du processus d'apprentissage, grâce à différents résultats positifs obtenus suite aux travaux d'ANGLUIN (décrits dans le chapitre 2). Cependant, ces résultats restent très éloignés de la problématique de l'apprentissage automatique des langues naturelles : ils sont pour la plupart exprimés en termes de propriétés formelles des langages considérés, à un niveau d'abstraction élevé qui en rend difficile l'application à des formalismes grammaticaux concrets. De ce point de vue, le plus appliqué de ces travaux est sans doute celui de SHINOHARA, qui montre dans [95] l'apprenabilité des grammaires contextuelles d'au plus k règles. Si l'expressivité de ces dernières est suffisante pour les langues naturelles, leur intérêt linguistique au niveau de la représentation des langues est tout de même limité.

En 1998, KANAZAWA revisite dans [60] l'algorithme d'apprentissage de grammaires AB rigides (à partir de structures) proposé par BUSZKOWSKI dans [27], et en tire de nombreuses conséquences pour

l'apprenabilité des grammaires AB dans le modèle de GOLD. Les apports de KANAZAWA sont multiples : du point de vue de l'inférence grammaticale, il montre de nouveaux résultats positifs et développe pour cela de nouvelles techniques de preuve. Mais surtout, ses résultats sont les premiers pour le modèle de GOLD à traiter d'un formalisme grammatical présentant de bonnes prédispositions à la représentation des langues naturelles, à savoir les grammaires catégorielles. C'est la raison pour laquelle son travail constitue une partie très importante de la base des travaux présentés dans les chapitres qui suivront.

Dans ce chapitre, nous présentons principalement ces résultats obtenus par KANAZAWA avec les grammaires AB. Nous commencerons par présenter l'algorithme RG de BUSZKOWSKI, dont le principe de fonctionnement est essentiel, aussi bien pour d'autres algorithmes qui en sont des variantes pour d'autres classes de grammaires, que pour montrer l'apprenabilité de ces classes. Nous donnons bien sûr aussi la preuve de convergence issue de [60] pour cet algorithme. Ensuite viendront les résultats d'élasticité finie pour certaines sous-classes des grammaires AB, qui constituent le cœur des contributions de KANAZAWA. Dans la dernière partie nous présentons les principales conséquences obtenues à partir du cas des grammaires AB pour quelques autres formes de grammaires catégorielles. Ceci concerne notamment une généralisation à ce que KANAZAWA nomme les *grammaires combinatoires générales*, dont on verra tout l'intérêt dans les prochains chapitres.

3.2 Apprentissage de grammaires AB rigides à partir de structures

L'algorithme RG (pour *Rigid Grammars*), proposé par Buszkowski [27], apprend la classe des grammaires AB rigides à partir de FA-structures (au sens du modèle de Gold).

3.2.1 Algorithme

L'algorithme RG comporte deux étapes. La première consiste à construire une *grammaire générale* à partir de la séquence de FA-structures fournies comme exemples :

Soit $D = \langle T_1, .., T_n \rangle$ la séquence de FA-structures en entrée.

1. Étiquetage d'une structure T_i :
   (a) la racine de T_i est étiquetée par le type S (le type primitif qui caractérise les phrases correctes) ;
   (b) en allant de la racine vers les feuilles, les fils de chaque nœud t sont étiquetés de la manière suivante : une nouvelle variable x est créée, avec laquelle le nœud argument est étiqueté. L'autre nœud est étiqueté t/x ou $x\backslash t$ selon qu'il s'agit d'un nœud FA ou BA. Le nœud argument est celui de la branche droite s'il s'agit d'un nœud FA, gauche si c'est un nœud BA.
2. Soit $\langle P_1..P_n \rangle$ l'ensemble des arbres de dérivation construits par étiquetage des FA-structures $\langle T_1..T_n \rangle$. Pour chaque arbre P_i et chaque feuille de cet arbre, une règle $w \mapsto t$ est créée, où w est le mot correspondant à cette feuille et t le type obtenu après étiquetage pour cette feuille. La grammaire ainsi obtenue est la grammaire générale $GF(D)$.

De fait, la phase d'étiquetage se résume à rassembler toutes les contraintes données par les exemples dans une grammaire unique. Par construction, chaque FA-structure de l'ensemble donné en entrée appartient au langage de structures de cette grammaire (et par conséquent le produit de la structure appartient au langage de chaînes de la grammaire).

Mais comme chaque occurence d'un mot dans un exemple se traduit par l'ajout d'une nouvelle règle, il est évident que cette seule étape ne permet pas d'apprendre le langage décrit, puisque la taille de la grammaire augmente indéfiniment. C'est la raison pour laquelle la seconde étape *d'unification* doit transformer $GF(D)$ en une grammaire rigide :

1. Pour chaque mot w défini dans $GF(D)$, soit A_w l'ensemble des types associés au mot w. Soit $\mathcal{A}$ la famille composée de tous les ensembles A_w. Soit σ_u *l'unifieur le plus général* (MGU) de $\mathcal{A}$ (voir partie 1.2.2.2).
2. On définit la grammaire $RG(D)$ par $RG(D) = \sigma_u[GF(D)]$: par définition, toute règle $w \mapsto t$ de $GF(D)$ est remplacée par $w \mapsto \sigma_u(t)$ dans $RG(D)$. Comme tous les types d'un même mot ont été unifiés, la grammaire obtenue est rigide (par exemple, si $GF(D)$ contenait des règles $w \triangleright t_1$ et $w \triangleright t_2$, par définition de σ_u on a $\sigma_u(t_1) = \sigma_u(t_2)$).

Kanazawa donne dans [60] une preuve de convergence pour cet algorithme (résumée ci-après dans la partie 3.2.3), démontrant ainsi que les langages de FA-structures de grammaires AB rigides sont apprenables (ce qu'on simplifie dans la littérature du domaine en "les grammaires AB rigides sont apprenables à partir de structures").

Cet algorithme a quelques qualités intéressantes : il est tout d'abord efficace, puisque sa complexité algorithmique (en fonction de la taille des exemples) est seulement quadratique[1]. Il peut être utilisé de manière incrémentale, ainsi il n'est pas nécessaire de recalculer la grammaire à partir de l'ensemble des phrases à chaque nouvel exemple. Enfin il produit une unique grammaire solution (la plus générale) quel que soit l'ensemble d'exemples proposés.

Dans la mesure où les FA-structures se résument à un parenthésage en constituants enrichi d'une information d'orientation (qu'on peut assimiler au sens d'une dépendance entre constituants), elles sont un intermédiaire assez équilibré entre les simples chaînes (phrases) du langage et leurs arbres de dérivation complets. L'ajout de ces informations structurelles pour l'apprentissage est généralement justifié par le fait que l'enfant acquérant un langage dispose dans son environnement "d'informations complémentaires" qui l'aident dans son apprentissage. Étant donné l'extrême précision des FA-structures par rapport à la grammaire, il est cependant discutable de considérer que l'enfant a à sa disposition de telles informations.

[1] Il ne faut pas confondre la complexité de l'algorithme d'apprentissage et la complexité de l'apprentissage en lui-même : la complexité de l'algorithme est le nombre de calculs élémentaires (au sens classique, c'est-à-dire dans une représentation raisonnable telle qu'une machine de Turing) nécessaire pour calculer la grammaire, en fonction de la taille de l'ensemble d'exemples (on choisit un critère de taille raisonnable, typiquement la somme du nombre de mots de chaque phrase). Mais la grammaire ainsi calculée n'est pas nécessairement la grammaire cible, puisque le modèle ne donne aucun moyen de prédire le nombre d'exemples nécessaires pour l'atteindre. Par conséquent on peut imaginer des cas où il faut un nombre exponentiel d'exemples (en fonction de la taille du vocabulaire par exemple) pour réaliser l'apprentissage : la complexité du processus global d'apprentissage n'est pas calculable a priori (sauf cas particuliers).

3.2.2 Exemple

On considère l'ensemble D de FA-structures représentées (après étiquetage des nœuds) sur la figure 3.1. La phase d'étiquetage permet d'obtenir la grammaire générale $GF(D)$ suivante :

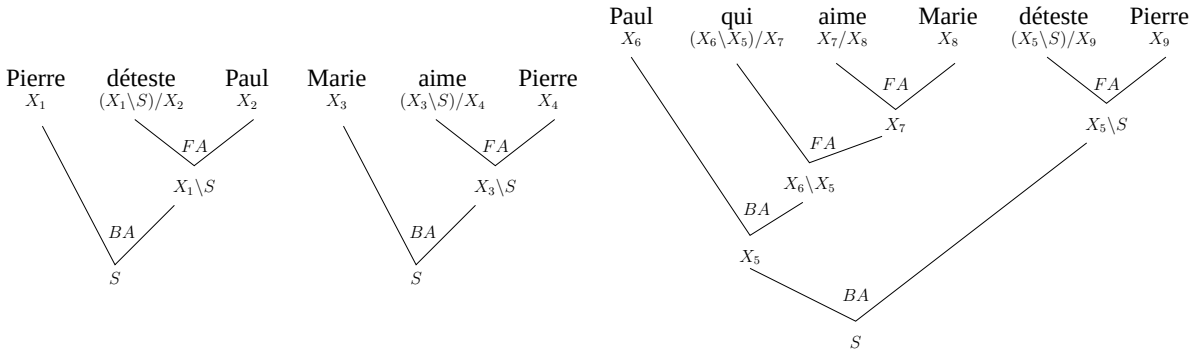


Figure 3.1 – FA-structures après étiquetage

$$GF(D) = \begin{cases} \text{Pierre} & \mapsto & X_1, X_4, X_9 \\ \text{Marie} & \mapsto & X_3, X_8 \\ \text{Paul} & \mapsto & X_2, X_6 \\ \text{aime} & \mapsto & (X_3\backslash S)/X_4, X_7/X_8 \\ \text{déteste} & \mapsto & (X_1\backslash S)/X_2, (X_5\backslash S)/X_9 \\ \text{qui} & \mapsto & (X_6\backslash X_5)/X_7 \end{cases}$$

Le calcul du MGU unifie les types suivants :

$$\begin{cases} X_1 = X_4 = X_9, \\ X_3 = X_8, \\ X_2 = X_6, \\ X_7 = (X_3\backslash S), \\ X_4 = X_8, \\ X_1 = X_5, \\ X_2 = X_9 \end{cases}$$

Ce qui donne : $X_1 = X_2 = X_3 = X_4 = X_5 = X_6 = X_8 = X_9$ et $X_7 = X_1\backslash S$. Donc la grammaire $RG(D)$ est :

$$RG(D) = \begin{cases} \text{Pierre} & \mapsto & X_1 \\ \text{Marie} & \mapsto & X_1 \\ \text{Paul} & \mapsto & X_1 \\ \text{aime} & \mapsto & (X_1\backslash S)/X_1 \\ \text{déteste} & \mapsto & (X_1\backslash S)/X_1 \\ \text{qui} & \mapsto & (X_1\backslash X_1)/(X_1\backslash S) \end{cases}$$

3.2.3 Preuve de convergence de l'algorithme RG

KANAZAWA démontre dans [60] que l'algorithme RG converge, c'est-à-dire que pour toute grammaire AB rigide G et toute énumération de FA-structures appartenant au langage de cette grammaire, il existe une étape dans l'énumération à partir de laquelle l'algorithme renvoie toujours une grammaire G' telle que $FL(G') = FL(G)$. Nous donnons ci-dessous les grandes lignes de cette démonstration (on donnera dans la partie 4.4 la version généralisée de cette démontration, avec plus de détails).

Définition 3.1 (Substitution sur une grammaire)**.** Étant donné une grammaire AB G, la grammaire $\sigma[G]$ est définie par $w \triangleright_{\sigma[G]} \sigma(t)$ si et seulement si $w \triangleright_G t$.

Proposition 3.1 (Buszkowski et Penn)**.** *Si $\sigma[G_1] \subseteq G_2$ alors $FL(G_1) \subseteq FL(G_2)$.*

Démonstration. Cas particulier de la proposition 4.11 (p. 112). □

Définition 3.2 (Substitution fidèle)**.** Une substitution σ est *fidèle* à une grammaire $G = \langle \Sigma, Pr, \triangleright \rangle$ si pour tout $w \in \Sigma$ tel que $w \triangleright t_1$ et $w \triangleright t_2$, si $t_1 \neq t_2$ alors $\sigma(t_1) \neq \sigma(t_2)$.

Définition 3.3 ($\sqsubseteq$)**.** Soit $\sqsubseteq$ la relation binaire sur les grammaires définie par $G \sqsubseteq G'$ si et seulement s'il existe une substitution σ telle que

- σ est fidèle à G ;
- $\sigma[G] \subseteq G'$.

Définition 3.4. La taille d'une grammaire G, notée $\|G\|$, est définie par

$$\|G\| = \sum_{w \triangleright t \in G} \|t\|.$$

Proposition 3.2. *Si $G_1 \sqsubseteq G_2$ alors $\|G_1\| \leq \|G_2\|$.*

Démonstration. Cas particulier de la proposition 4.15 (p. 114). □

Corollaire 3.1. *L'ensemble $\{ G' \mid G' \sqsubseteq G \}$ est fini (aux renommages près[2]).*

Démonstration. Cas particulier de la proposition 4.2 (p. 114). □

Proposition 3.3. *Quel que soit D un ensemble fini de FA-structures, $D \subseteq FL(GF(D))$.*

Démonstration. Par construction, l'étiquetage d'une FA-structure T vérifie les contraintes données par les règles FA et BA, donc est conforme à la FA-structure. La grammaire $GF(D)$ étant simplement constituée des types ainsi créés, il est clair que la structure obtenue par étiquetage est un arbre de dérivation pour la grammaire $GF(D)$ et que cet arbre est une instance de T, donc $D \subseteq FL(GF(D))$. □

Proposition 3.4. *Soit G une grammaire AB et D un ensemble fini de FA-structures.*
$D \subseteq FL(G)$ si et seulement s'il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$.

[2] C'est-à-dire si on ne distingue pas G_1 et G_2 lorsque $G_1 \sqsubseteq G_2$ et $G_2 \sqsubseteq G_1$.

Démonstration.

$\Rightarrow$. Soit $D = \{T_1, \ldots, T_n\}$. Pour tout i, $T_i \in FL(G)$ donc il existe une instance P_i de T_i pour G. Par construction, il existe une substitution σ_i telle que $P_i = \sigma_i[P'_i]$, avec P'_i l'instance de T_i pour $GF(D)$ (voir démonstration détaillée de l'existence d'une telle substitution dans la proposition 4.18, dont le cas présent est un cas particulier). Autrement dit à tout nœud étiqueté x dans P'_i correspond un nœud étiqueté $\sigma_i(x)$ dans P_i. Soit σ la substitution définie comme l'union de tous les σ_i : σ existe car les ensembles de variables créés dans l'algorithme $GF(D)$ sont disjoint pour chaque FA-structure T_i. Ainsi par définition à toute règle $w \triangleright t$ dans $GF(D)$ correspond une règle $w \triangleright \sigma(t)$ dans G, ce qui implique que $\sigma[GF(D)] \subseteq G$.

$\Leftarrow$. $\sigma[GF(D)] \subseteq G$ donc $FL(GF(D)) \subseteq FL(G)$ selon la proposition 3.1. Or $D \subseteq FL(GF(D))$ d'après la proposition 3.3, donc $D \subseteq FL(G)$. □

Proposition 3.5. *Soit G une grammaire AB rigide et D un ensemble fini de FA-structures.*
$D \subseteq FL(G)$ si et seulement si $RG(D)$ existe et $RG(D) \sqsubseteq G$.

Démonstration.

$\Rightarrow$. D'après la proposition 3.4, il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$. Or G est rigide, donc $\sigma[GF(D)]$ l'est aussi. Par conséquent σ est un unifieur de la famille d'ensembles $\mathcal{A} = \{ A_w \mid w \in \Sigma \}$, avec $A_w = \{ t \mid (w \triangleright t) \in GF(D) \}$. L'existence d'un unifieur pour $\mathcal{A}$ garantit l'existence du plus général unifieur σ_{mgu} de cette famille (proposition 1.1). Ainsi $RG(D) = \sigma_{mgu}[GF(D)]$ existe, et par définition du MGU il existe une substitution τ telle que $\sigma = \tau \circ \sigma_{mgu}$. On a donc $\tau[RG(D)] = \tau[\sigma_{mgu}[GF(D)]] = \sigma[GF(D)]$. Or $\sigma[GF(D)] \subseteq G$, donc $\tau[RG(D)] \subseteq G$. τ est fidèle puisque $RG(D)$ est une grammaire rigide, donc $RG(D) \sqsubseteq G$.

$\Leftarrow$. Par définition $RG(D) \sqsubseteq G$ implique qu'il existe une substitution σ telle que $\sigma[RG(D)] \subseteq G$. Or $RG(D) = \sigma_{mgu}[GF(D)]$ (par définition), donc $\sigma[\sigma_{mgu}[GF(D)] \subseteq G$. D'après la proposition 3.4 l'existence d'une telle substitution ($\sigma \circ \sigma_{mgu}$) implique que $D \subseteq FL(G)$.

□

Proposition 3.6. *Soit G une grammaire AB rigide, et D_1, D_2 deux ensembles finis de FA-structures.*
Si $D_1 \subseteq D_2 \subseteq FL(G)$ alors $RG(D_1)$ et $RG(D_2)$ existent, et $RG(D_1) \sqsubseteq RG(D_2) \sqsubseteq G$.

Démonstration. Il découle de la proposition 3.5 que $RG(D_1)$ et $RG(D_2)$ existent, et que $RG(D_1) \sqsubseteq G$ et $RG(D_2) \sqsubseteq G$. $RG(D_2) \sqsubseteq RG(D_2)$ car $\sqsubseteq$ est réflexive, donc selon la proposition 3.5 (utilisée ici dans l'autre sens) on a $D_2 \subseteq FL(RG(D_2))$. Comme $D_1 \subseteq D_2 \subseteq FL(RG(D_2))$, on obtient que $RG(D_1) \sqsubseteq RG(D_2)$ en appliquant encore une fois la proposition 3.5. □

Théorème 3.1. *L'algorithme RG apprend la classe des langages de grammaires AB rigides à partir de structures.*

Démonstration. Soit $\{T_1, T_2, \ldots\}$ une énumération de $FL(G)$, et soit $D_i = \{T_1, T_2, \ldots, T_i\}$ pour tout $i \in \mathbb{N}$. Par hypothèse, $D_i \subseteq D_{i+1} \subseteq FL(G)$ pour tout $i \in \mathbb{N}$. D'après la proposition 3.6, ceci implique que $RG(D_i)$ et $RG(D_{i+1})$ existent, et que $RG(D_i) \sqsubseteq RG(D_{i+1}) \sqsubseteq G$. Or il n'y a qu'un nombre fini

de grammaires G' différentes telles que $G' \sqsubseteq G$ (par le corollaire 3.1), c'est pourquoi RG converge nécessairement sur la séquence $\langle T_i \rangle_{i \in \mathbb{N}}$ vers une grammaire G_n : il existe $n \in \mathbb{N}$ tel que $RG(D_i) = G_n$ pour tout $i \geq n$. Ainsi $FL(G) = \{ T_i \mid i \in \mathbb{N} \} \subseteq FL(G_n)$, car $D_i \subseteq FL(RG(D_i)) = FL(G_n)$ pour tout $i \geq n$ (par la proposition 3.5, utilisée avec $RG(D_i) \sqsubseteq RG(D_i)$). Comme $G_n = RG(D_n) \sqsubseteq G$, on a aussi $FL(G_n) \subseteq FL(G)$ d'après la proposition 4.11. Par conséquent on a $FL(G_n) \subseteq FL(G)$ et $FL(G) \subseteq FL(G_n)$, d'où $FL(G_n) = FL(G)$.

On a donc montré que pour toute grammaire AB rigide G et toute énumération $\langle T_1, T_2, \ldots \rangle$ de $FL(G)$, il existe un entier $n \in \mathbb{N}$ tel que $FL(RG(\{T_1, T_2, \ldots, T_i\})) = FL(G)$ pour tout entier $i \geq n$. En d'autres termes l'algorithme RG apprend la classe des langages de grammaires AB rigides à partir de structures.

□

3.2.4 Discussion

KANAZAWA a donc démontré à partir de l'algorithme RG de BUSZKOWSKI que non seulement la classe des langages de FA-structures des grammaires AB est apprenable, mais aussi qu'il existe un algorithme incrémental efficace (de complexité algorithmique polynomiale) pour mener à bien cet apprentissage. Dans la mesure où les grammaires AB sont un formalisme grammatical qui constitue la base de systèmes relativement adaptés aux langues naturelles, ce résultat représente un premier pas notable dans la direction de l'apprentissage automatique des langues naturelles.

Cependant ce résultat a plusieurs limitations très fortes : d'une part, les grammaires AB en tant que telles ne sont pas un formalisme suffisant pour les langues naturelles. Il sera donc nécessaire d'étudier si la méthode utilisée dans ce cas est extensible à des formalismes plus pertinents, permettant de gérer avec plus de finesse les subtilités des langues naturelles. D'autre part, même si l'on considère les grammaires AB comme un formalisme d'un certain intérêt pour les langues naturelles, ce résultat comporte deux limites qui en réduisent considérablement l'intérêt : le fait que des FA-structures soient nécessaires en entrée et non de simples phrases, et le fait que seules les grammaires rigides puissent être prises en compte.

La question du type d'entrées de l'algorithme d'apprentissage peut être analysée de deux points de vues distincts. Si l'on considère que l'objectif de l'algorithme est de modéliser l'acquisition humaine du langage, la présence d'informations supplémentaires par rapport à la simple phrase peut être justifiée : l'enfant qui apprend sa langue dispose effectivement d'un ensemble d'éléments extérieurs à la phrase elle-même, dont on peut supposer qu'ils lui sont utiles (même au niveau de la décomposition syntaxique de la phrase). Cependant la forme prise par cette information au niveau des FA-structures est contestable : le parenthésage en constituants sans ambiguité est déjà un élément d'une (trop ?) grande précision, quant à l'étiquetage des nœuds en FA/BA, interprétable comme l'orientation d'une dépendance entre constituants, il s'agit d'une information extrêmement spécifique aux grammaires AB, et on peut donc douter de la pertinence d'une telle information sur un plan linguistique. Le problème est différent si l'on considère que l'algorithme doit être un outil pratique utile pour découvrir automatiquement la syntaxe des langues : la question qui se pose alors est celle de l'existence de ressources aussi spécifiques que des FA-

structures de grammaires AB sur des volumes de données suffisamment vastes pour décrire une langue. Or là encore la réponse est négative : non seulement il existe très peu de données formalisées en termes de grammaires AB, mais encore moins avec des FA-structures. Et ce d'autant plus que que les FA-structures sont liées très fortement à la forme de la grammaire à laquelle elles appartiennent (ce qui est par ailleurs le principe même de l'algorithme RG). Ainsi la création d'un corpus de FA-structures sur une langue donnée présuppose une connaissance très précise de la grammaire AB sous-jacente, ce qui ôte en grande partie l'intérêt de "ré-apprendre" une telle grammaire.

Le problème de la restriction aux grammaire rigides est également quasiment rédhibitoire lorsqu'on s'intéresse aux langues naturelles. En effet, rappelons que cette limitation interdit à tout mot du vocabulaire de se voir attribuer plusieurs types syntaxiques différents. On voit tout de suite le problème que cela pose pour les homonymes. Par exemple, un mot tel que "montre" qui peut être aussi bien un verbe qu'un nom commun, ne peut se contenter d'une seule catégorie syntaxique : si c'était le cas, cela signifierait que les verbes et les noms se comportent syntaxiquement tous de manière identique, ce qui n'est clairement pas raisonnable. Le problème est plus grave encore pour les mots grammaticaux (déterminants, pronoms, prépositions, etc.) : ces mots peuvent très souvent être utilisés de différentes façons dans la phrase, correspondant chacune à une catégorie syntaxique différente. Par conséquent il semble très peu probable que les langues naturelles soient modélisables à l'aide de grammaires AB rigides, même de manière approximative.

3.3 Grammaires AB et élasticité finie

Le principal apport de KANAZAWA dans [60] est sans doute la démonstration de l'élasticité finie des langages de FA-structures de grammaires AB rigides, et les conséquences qui en découlent quant à l'apprenabilité des grammaires k-valuées d'une part et des langages de chaînes d'autre part. Outre l'intérêt du résultat en lui-même, KANAZAWA proposait en quelque sorte à l'aide de son théorème sur la relation finiment valuée (voir proposition 2.14) une nouvelle méthode de preuve d'apprenabilité par extension de l'élasticité finie. Cette méthode a été depuis largement réutilisée, et nous en ferons nous-même usage dans les prochains chapitres.

3.3.1 Élasticité finie des langages de structures rigides

Nous donnons dans cette partie un aperçu de la démonstration fournie par KANAZAWA de l'élasticité finie des langages de FA-structures des grammaires AB rigides. Les bases nécessaires à cette démonstration occupant quasiment tout un chapitre de [60], nous ne donnons pas les preuves complètes de tous les lemmes utilisés, mais seulement de la principale proposition (lemme clé 5.56 dans [60], p. 76). Toutefois on trouvera tous les éléments de preuve manquants dans le chapitre 6, qui contient une démonstration complète de l'élasticité finie d'une classe de langages englobant celle des grammaires AB, basée sur celle de KANAZAWA mais plus complexe.

Nous synthétisons ci-dessous quelques définitions utilisées par KANAZAWA. Les concepts ci-dessous seront généralisés et détaillés dans le chapitre 6.

Définition 3.5.

- Une grammaire G ne contient pas de type inutile si pour tout type $t \in STLex(G)$ il existe un arbre de dérivation complet (de racine s) pour G tel que t est l'un des nœuds de cet arbre[3].
- $Lex(G) = \{ t \in Tp \mid w \triangleright_G t \}$.
- $STLex(G) = \{ t \in Tp \mid$ il existe $t' \in Lex(G)$ tel que t est un sous-type de $t' \}$.
- $Var(G) = \{ t \in STLex(G) \mid t \in Pr$ et $t \neq s \}$.
- $head(t)$ est la fonction définie inductivement par $head(t) = t$ si t est un type primitif et $head(A/B) = head(B\backslash A) = head(A)$ sinon.
- $Heads(G) = \{ head(t) \mid t \in Lex(G) \}$.
- $args^*(t)$ est la fonction définie inductivement par $args^*(t) = \emptyset$ si t est primitif et $args^*(A/B) = args^*(B\backslash A) = args^*(A) \cup \{B\}$.
- Deux grammaires sont dites équivalentes, ce qui est noté $G \equiv G'$, si elles sont identiques modulo un renommage de variables.

Proposition 3.7. *Si G ne contient pas de types inutiles alors $Heads(G) = Var(G) \cup \{s\}$.*

Proposition 3.8. *Soit σ une substitution. Si u, t, t' sont des types tels que $t = \sigma(u)$ et $t' \in args^*(t)$, alors (au moins) une des deux conditions suivantes est vérifiée :*

- *il existe $u' \in args^*(u)$ tel que $\sigma(u') = t'$;*
- *ou $t' \in args^*(\sigma(head(u)))$.*

Démonstration. Voir la démonstration de la proposition 6.7 (p. 165), très similaire. □

Proposition 3.9. *S'il existe $x \in Heads(G)$ et $y \in STLex(G)$ tels que $y \in args^*(t)$ pour tout $t \in Lex(G)$ tel que $head(t) = x$, alors tout arbre de dérivation de racine x pour G contient strictement un sous-arbre de racine y.*

Démonstration. La forme des règles FA et BA indique clairement que tous les arguments de t doivent être "consommés" pour obtenir le type $head(t)$. □

Proposition 3.10. *Soient G et G' deux grammaires AB rigides sans types inutiles.*
Si $\sigma[G] = G'$ et $|Heads(G)| = |Heads(G')|$ alors σ est un renommage.

Démonstration. Soit $\{x_1, \ldots, x_n\} = Heads(G) \setminus \{s\}$, et soit $z_i = head(\sigma(x_i))$ pour tout i. Notons que $Heads(G') = \{z_1, \ldots, z_n\}$, avec $z_i \neq z_j$ pour tout $i \neq j$, car $|Heads(G)| = |Heads(G')|$. Nous allons montrer par l'absurde que $\sigma(x_i) = z_i$ pour tout i, $1 \leq i \leq n$. Pour cela nous commençons par montrer que s'il existe j, $1 \leq j \leq n$, tel que $\sigma(x_j) \neq z_j$ alors il existe aussi k, $1 \leq k \leq n$ et $k \neq j$ tel que $\sigma(x_k) \neq z_k$ et z_k dépend de z_j, c'est-à-dire que tout arbre de dérivation dont la racine est z_k contient nécessairement un sous-arbre de dérivation dont la racine est z_j.

Supposons que $z_j \neq \sigma(x_j)$. Par définition $z_j \neq z_i$ pour tout $i \neq j$, autrement dit $z_j \neq head(\sigma(x_i))$ et donc $z_j \neq \sigma(x_i)$ pour tout $i \neq j$ puisque z_j est primitif. Ainsi $z_j \neq \sigma(x_i)$ pour tout i, $1 \leq i \leq n$.

[3] Il s'agit ici d'une définition légèrement simplifiée par rapport à celle de [60].

G' ne contient pas de type inutile, donc il existe un arbre de dérivation complet contenant un sous-arbre dont la racine est z_j. $z_j \neq s$ (sinon $|Heads(G')| < |Heads(G)|$), par conséquent il existe $t \in Lex(G')$ tel que $z_j \in args^*(t)$. Soit $z_k = head(t)$: $z_j \neq \sigma(x_i)$ pour tout i, donc $z_j \in args^*(\sigma(x_k))$ d'après le lemme 3.8. Ainsi $\sigma(x_k)$ n'est pas un type primitif, donc $z_k \neq \sigma(x_k)$. De plus $z_k \neq z_i$ pour tout $i \neq k$, donc tout type $t' \in Lex(G')$ tel que $head(t') = z_k$ vérifie $z_j \in args^*(t')$, ce qui signifie que z_k dépend de z_j (selon le lemme 3.9).

Par conséquent l'hypothèse qu'il existe un j, $1 \leq j \leq n$, tel que $\sigma(x_j) \neq z_j$ entraîne le fait qu'il existe une séquence infinie $z_{j_1}, z_{j_2}, \ldots$ telle que $z_{j_{i+1}}$ dépend de z_{j_i}. Or $|Heads(G')| \leq |\Sigma|$, donc il existe nécessairement k et k' tels que $z_{j_k} = z_{j'_k}$, c'est-à-dire qu'il existe un type primitif dans $Heads(G')$ qui dépend de lui-même. Par hypothèse G' ne contient pas de types inutiles, donc ceci est une contradiction.

Ainsi $\sigma(x_i) = z_i$ pour tout i ($1 \leq i \leq n$), donc tout type primitif de $Heads(G)$ est transformé en type primitif par σ. Comme $|Heads(G)| = |Heads(G')|$, cela implique que σ est un simple renommage de variables.

□

Corollaire 3.2. *Il n'existe pas de séquence infinie $\langle G_i \rangle_{i \in \mathbb{N}}$ de grammaires AB rigides sans types inutiles sur un vocabulaire Σ telle que $G_i \sqsubseteq G_{i+1}$ et $G_i \not\equiv G_{i+1}$ pour tout $i \in \mathbb{N}$.*

Démonstration. D'après le lemme 3.7, $|Var(G)| < |Heads(G)|$, or $|Heads(G)| \leq |\Sigma|$ pour toute grammaire AB rigide sans types inutiles. De plus $\sigma[G_i] = G_{i+1}$ avec $|Heads(G_i)| = |Heads(G_{i+1})|$ implique $G_i \equiv G_{i+1}$ selon la proposition 3.10, donc toute séquence $G'_1, \ldots, G'_n$ de grammaires AB rigides sans types inutiles telles que $\sigma[G'_i] = G'_{i+1}$ et $G'_i \not\equiv G'_{i+1}$ vérifie $n \leq |\Sigma|$.

Ainsi dans toute séquence $G''_1, \ldots, G''_m$ de grammaires AB rigides sans types inutiles telles que $\sigma[G''_i] \subseteq G''_{i+1}$ et $G''_i \not\equiv G''_{i+1}$ il ne peut y avoir au plus que n grammaires consécutives telles que $\sigma[G''_i] = G''_{i+1}$. De plus $|G''_i| \leq |\Sigma|$ puisque les grammaires sont rigides, et $|G''_i| < |G''_{i+1}|$ si $\sigma[G''_i] \subset G''_{i+1}$, donc une telle séquence de grammaires vérifie $m \leq |\Sigma|^2$. Par conséquent Il n'existe pas de séquence infinie $\langle G_i \rangle_{i \in \mathbb{N}}$ de grammaires AB rigides sans types inutiles sur un vocabulaire Σ telle que $G_i \sqsubseteq G_{i+1}$ et $G_i \not\equiv G_{i+1}$ pour tout $i \in \mathbb{N}$. □

Remarque : La borne $|\Sigma|^2$ que nous proposons dans cette démonstration n'est pas optimale : en fait KANAZAWA a montré dans [60] qu'une telle séquence de grammaires a pour longueur maximum $|\Sigma|$. Cette simplification de la démonstration n'a pas d'incidence sur l'élasticité finie de la classe de langages, dont la preuve est l'objectif de cette partie.

Proposition 3.11 (KANAZAWA)**.**
La classe des langages de FA-structures des grammaires AB rigides a l'élasticité finie.

Démonstration. Supposons qu'il existe une séquence infinie de FA-structures $T_0, T_1, T_2, \ldots$ et une séquence infinie de grammaires AB rigides $G_1, G_2, \ldots$ telles que

$$\{T_0, \ldots, T_{i-1}\} \subseteq FL(G_i) \text{ et } T_i \notin FL(G_i) \text{ pour tout } i > 0.$$

D'après la proposition 3.5, si $\{T_0, \ldots, T_{i-1}\} \subseteq FL(G_i)$ alors $RG(\{T_0, \ldots, T_{i-1}\})$ existe et on a $RG(\{T_0, \ldots, T_{i-1}\}) \sqsubseteq G_i$ pour tout $i > 0$. De même la proposition 3.5 implique que pour tout i $T_i \notin RG(\{T_0, \ldots, T_{i-1}\})$, d'où le fait que $RG(\{T_0, \ldots, T_{i-1}\}) \not\equiv RG(\{T_0, \ldots, T_i\})$. Or comme la proposition 3.6 donne $RG(\{T_0, \ldots, T_{i-1}\}) \sqsubseteq RG(\{T_0, \ldots, T_i\})$, on obtient que $RG(\{T_0, \ldots, T_{i-1}\}) \sqsubset RG(\{T_0, \ldots, T_i\})$ pour tout $i > 0$. Par conséquent il existe une séquence infinie de grammaires AB rigides $G'_1, G'_2, \ldots$ sans types inutiles[4], avec $G'_i = RG(\{T_0, \ldots, T_{i-1}\})$ pour tout $i > 0$, telle que $G'_i \sqsubset G'_{i+1}$: ceci contredit le corollaire 3.2, donc la classe des langages de FA-structures des grammaires AB rigides a l'élasticité finie. □

3.3.2 Extension de l'élasticité finie aux langages de chaînes de grammaires AB k-valuées

L'intérêt de la propriété d'élasticité finie sur les langages de FA-structures des grammaires AB rigides réside dans l'extention qu'en a faite KANAZAWA aux grammaires k-valuées d'une part et aux langages de chaînes d'autre part.

3.3.2.1 Élasticité finie des langages de structures des grammaires AB k-valuées

Afin d'utiliser la proposition 2.14, KANAZAWA définit une relation finiment valuée entre les langages (de FA-structures) des grammaires AB rigides et k-valuées sur un vocabulaire Σ. Dans la suite on considère que $k > 1$ est un entier fixé. Soit Σ' le vocabulaire défini par

$$\Sigma' = \bigcup_{w \in \Sigma} \{w_1, \ldots, w_k\},$$

où tous les w_i sont distincts. Comme on peut le voir Σ' est un vocabulaire comportant $k \times |\Sigma|$ mots différents, chaque mot w de Σ pouvant être associé à k copies de w dans Σ'.

Soit $\prec$ un ordre total sur les types : pour toute grammaire k-valuée G et tout $t \in Lex(G)$, soit $f(w, i) = t$ avec t le $i$ème type de l'ensemble $\{t' | w \triangleright_G t'\}$ ordonné selon $\prec$. Soit $rc(G)$ la grammaire rigide sur Σ' définie par

$$w_i \triangleright_{rc(G)} f(w, i) \text{ pour tout } w \in \Sigma \text{ et tout } i, 1 \leq i \leq k.$$

On note que la fonction rc est bijective. Soit $amb : \mathcal{FL}_{\Sigma'} \mapsto \mathcal{FL}_{\Sigma}$ la fonction définie par :

- $amb(w_i) = w$ pour tout $w \in \Sigma$;
- $amb(fa(T, T')) = fa(amb(T), amb(T'))$ pour toutes FA-structures T et T' sur Σ' ;
- $amb(ba(T, T')) = ba(amb(T), amb(T'))$ pour toutes FA-structures T et T' sur Σ' ;

amb est classiquement étendue aux ensembles de FA-structures de la manière suivantes : $amb[D] = \{\ amb(T) \mid T \in D\ \}$. Par définition, on a $FL(G) = amb[FL(rc(G))]$ pour toute grammaire k-valuée G.

[4] $RG(D)$ ne contient pas de types inutiles par construction. Nous ne donnons pas de détail sur ce point dans ce chapitre mais dans le chapitre 6, pour la version généralisée.

Proposition 3.12. *La classe des langages de FA-structures des grammaires AB k-valuées a l'élasticité finie.*

Démonstration. Selon la proposition 3.11, la classe $\{ FL(G) \mid G$ est une grammaire rigide sur $\Sigma' \}$ a l'élasticité finie. Ainsi la classe $\{ FL(rc(G)) \mid G$ est une grammaire k-valuée sur $\Sigma \}$ a aussi l'élasticité finie puisqu'il s'agit d'un sous-ensemble de la première classe. Il est clair que la relation amb entre $\mathcal{FL}_{\Sigma'}$ et $\mathcal{FL}_{\Sigma}$ est finiment valuée, donc selon la proposition 2.14 la classe

$$\{ amb[FL(rc(G))] \mid G \text{ est une grammaire } k\text{-valuée sur } \Sigma \}$$

a également l'élasticité finie. Cette classe est identique à la classe

$$\{ FL(G) \mid G \text{ est une grammaire } k\text{-valuée sur } \Sigma \},$$

ce qui signifie que la classe des langages de FA-structures des grammaires AB k-valuées a l'élasticité finie.

□

La différence entre le cas des grammaires rigides et celui des grammaires k-valuées ne se situe donc pas au niveau de l'apprenabilité théorique, mais au niveau de la réalisation pratique d'un tel apprentissage. En effet l'algorithme proposé par KANAZAWA dans ce cas consiste simplement à calculer toutes les possibilités d'unifier ou non un type donné avec l'un ou l'autre des k types possibles du mot. Il s'agit en fait de calculer tous les k-partitionnements possibles au lieu d'un unique MGU (voir la partie 4.4.4 pour un algorithme généralisé utilisant cette technique), dont le nombre est bien entendu exponentiel. COSTA FLORÊNCIO montrera ensuite que ce problème est NP-difficile [33]), ce qui signifie que l'algorithme exponentiel de KANAZAWA ne peut pas être amélioré de façon significative.

3.3.2.2 Élasticité finie des langages de chaînes des grammaires AB k-valuées

Proposition 3.13. *La classe des langages de chaînes des grammaires AB k-valuées a l'élasticité finie.*

Démonstration. Soit $\mathcal{G}_k$ l'ensemble des grammaires AB k-valuées. Rappelons que par définition $L(G) = \{ prod(T) \mid T \in FL(G) \}$ (où $prod(T)$ est le produit de la FA-structure T). Soit R la relation définie sur $\Sigma^* \times \mathcal{FL}$ par $x\ R\ T$ si et seulement si $x \in prod(T)$ (autrement dit x est la phrase représentée par la FA-structure T) : on remarque que pour toute grammaire G $R^{-1}[FL(G)] = \{ x \in \Sigma^* \mid \exists T \in FL(G)$ tel que $x \in prod(T) \} = L(G)$. Soit $G \in \mathcal{G}_k$ une $\mathcal{R}$-grammaire k-bornée. Pour tout x appartenant à $L(G)$ le nombre de FA-structures T telles que $x \in prod(T)$ est borné : en effet, si $|x| = n$, une telle structure contient n feuilles et $n-1$ nœuds de règles FA/BA. La relation R est donc finiment valuée, or d'après la proposition 3.12 pour tout $k \geq 0$ la classe $\{ FL(G) \mid G \in \mathcal{G}_k \}$ a l'élasticité finie. Par conséquent, d'après la proposition 2.14, la classe $\{ L(G) \mid G \in \mathcal{G}_k \}$ a aussi l'élasticité finie. □

Étant donné une phrase de taille n, il n'existe qu'un nombre fini de FA-structures qui peuvent lui correspondre : il y a "seulement" $\frac{(2n+2)!}{(n+1)!(n+2)!}$ parenthésages possibles (nombres de Catalan), et le

nombre d'étiquetages possibles des nœuds par FA/BA est de 2^{n-1}. L'algorithme proposé par KANAZAWA consiste simplement à calculer toutes les grammaires possibles en fonction de toutes les FA-structures possibles, ce qui est bien sûr de complexité exponentielle. COSTA FLORÊNCIO montrera ensuite dans [34] qu'il s'agit encore dans ce cas d'un problème NP-difficile. De plus, au niveau algorithmique, cette extension pose le problème de la recherche d'une grammaire minimale parmi l'ensemble de celles qui sont compatibles (voir [60]) : celui-ci est encore source d'inefficacité.

3.3.3 Discussion

Finalement dans ces deux cas l'extension à des classes de langages plus riches mène inévitablement à un problème d'efficacité, quasiment insoluble pour des applications réelles. Ceci illustre le fait que, même si l'élasticité finie est un critère très puissant pour démontrer l'apprenabilité « théorique » d'une classe de langages complexe à partir d'un cas simple, son utilisation ne préserve pas les propriétés algorithmiques du cas de base. Il est utile de noter que le problème des structures est très lié à la première partie de l'algorithme (construction de la grammaire générale), tandis que le problème de la rigidité des grammaires est lié à la seconde partie (l'unification). En fait il s'avère impossible de réaliser la première partie de l'algorithme en temps polynomial sans les FA-structures complètes, et de la même manière le déterminisme de la seconde partie est basé sur la rigidité des grammaires.

Sur le plan théorique il s'agit d'un résultat relativement intéressant pour les langues naturelles, mais la mise en pratique est évidemment impossible. NICOLAS a implémenté et testé l'algorithme de KANAZAWA pour ce cas, et obtient par exemple 126775 grammaires solutions en fixant seulement la valeur 2 à k, pour de petits exemples [81]. Le fait que l'algorithme ne soit plus déterministe ajoute encore la complication de choisir une grammaire parmi toutes les solutions, de manière à satisfaire la définition du modèle de Gold (une seule grammaire en sortie). KANAZAWA décrit les différents aspects de ce problème de la recherche d'une grammaire minimale dans l'ensemble de solutions, qui passe encore une fois par un algorithme exponentiel.

3.4 Autres résultats d'apprenabilité pour les grammaires catégorielles

À la suite des résultats d'apprenabilité positifs obtenus par KANAZAWA pour les grammaires catégorielles classiques, plusieurs travaux ont naturellement cherché à étendre ces résultats dans le champ des grammaires catégorielles. Cela inclut différents types d'études, en particulier sur les grammaires de LAMBEK, sur des formes différentes de données en entrée pour les grammaires AB, ainsi que différentes formes de généralisation.

3.4.1 Résultats d'apprenabilité sur des formalismes proches des grammaires AB

Comme on l'a vu dans la partie 1.3.2.2, les grammaires de LAMBEK sont une extension naturelle des grammaires AB présentant de nombreux avantages au niveau linguistique, notamment pour la représentation de la sémantique des phrases. Les questions relatives à l'apprenabilité des grammaires de

LAMBEK sont donc d'une grande importance dans la perspective de l'application du modèle de GOLD aux langues naturelles.

BONATO a proposé dans [21] un algorithme d'apprentissage des grammaires de LAMBEK rigides à partir de "phrases structurées". Dans [23], BONATO et RETORÉ reprennent ce résultat et l'étendent aux grammaires minimalistes. Cet algorithme est basé sur l'algorithme RG de Buszkowski, et la preuve de convergence dans le modèle de GOLD qui l'accompagne est inspirée de celle de KANAZAWA. Les données fournies en entrée à l'algorithme sont des arbres de preuve incomplets dont les nœuds sont étiquetés par les noms des règles applicables dans le calcul de LAMBEK, à l'instar des FA-structures pour l'algorithme RG. La complexité de l'algorithme est également polynomiale. Ce résultat positif est donc une extension fidèle de celui de BUSZKOWSKI et KANAZAWA pour le cas des grammaires AB rigides apprenables à partir de FA-structures.

Cependant les grammaires de LAMBEK ne possèdent pas les propriétés adéquates des grammaires AB pour l'élasticité finie. En fait la comparaison en termes d'apprenabilité avec les grammaires AB s'arrête au cas de l'apprentissage de grammaires rigides avec des structures : de nombreux résultats négatifs obtenus par LE NIR et FORET le prouvent [46],[47],[82]. En particulier, les grammaires de LAMBEK rigides classiques (non-associatives) ne sont pas apprenables à partir de chaînes. LE NIR et FORET [82] exhibent en effet le point limite suivant :

Exemple 3.1. Soient p et q des types primitifs et A_i le type défini pour tout $i \geq 0$ par $A_i = s$ si $i = 0$ et $A_i = (A_i/(p/p))/(q/q)$ sinon.

$G_i = \{\ a \triangleright p/p\ ;\ b \triangleright q/q\ ;\ c \triangleright A_i\ \}$

$G_* = \{\ a \triangleright p/p\ ;\ b \triangleright p/p\ ;\ c \triangleright s/(p/p)\ \}$

On peut montrer que $L(G_i) = c(b^*a^*)^i$, donc $L(G_i) \subset L(G_{i+1})$. De plus

$$\bigcup_{i \geq 0} L(G_i) = L(G_*) = c(b^*a^*)^* = c\{a, b\}^*,$$

par conséquent cette séquence de grammaires de LAMBEK rigides décrit un point limite, ce qui implique que les grammaires de LAMBEK rigides ne sont pas apprenables à partir de chaînes (voir théorème 2.1).

Les résultats de KANAZAWA, ainsi que les possibilités d'extension de la méthode d'apprentissage utilisée, ont donné lieu par la suite à de nombreux autres travaux. Dans le cadre des grammaires AB, DUDAU-SOFRONIE et TELLIER[100] [43] [42] adaptent les résultats de KANAZAWA sur les grammaires catégorielles de manière à prendre en compte des informations d'ordre sémantique dans les exemples, conformément à ce dont dispose un enfant dans son environnement. Dans le cadre plus large de l'inférence grammaticale, BESOMBES et MARION ont étudié l'utilisation d'algorithmes efficaces de type RG pour l'apprentissage de langages d'arbres réversibles et quelques applications [17] [19], dont certaines à des classes particulières de grammaires AB [18]. On peut également rattacher à cette branche de travaux issus de celui de KANAZAWA le résultat de BÉCHET sur l'apprenabilité des grammaires de liens k-valuées [14]. On retrouve en effet dans ce résultat un algorithme efficace pour le cas des gram-

maires rigides apprises à partir de structures, et une extension via l'élasticité finie au cas des grammaires k-valuées à partir de chaînes.

3.4.2 Généralisation : les grammaires combinatoires générales de KANAZAWA

Comme le souligne KANAZAWA dans le dernier chapitre de [60], on peut constater en observant les différents résultats qu'il a obtenus que le formalisme spécifique des grammaires AB n'y joue pas un rôle crucial. Il démontre en effet que l'algorithme RG peut être assez facilement adapté de façon à traiter de systèmes différents des grammaires AB, pourvu qu'ils soient à base de règles universelles applicables par substitution et qu'on y dispose de structures analogues aux FA-structures. En fait KANAZAWA présente cette partie de ces travaux comme une simple extension en direction des grammaires catégorielles combinatoires de STEEDMAN (voir la partie 1.3.2.3), mais nous verrons que l'on peut en envisager beaucoup d'autres applications.

Nous reprenons ici brièvement cette partie des travaux de KANAZAWA, ainsi que les extensions qui en sont faites. Comme ceux-ci constitueront la base d'une grande partie des extensions que nous proposerons dans les chapitres suivants, nous présentons ici seulement le principe de cette généralisation.

3.4.2.1 Cas des grammaires rigides à partir de structures (KANAZAWA)

Soit $\mathcal{O}$ un ensemble d'opérateurs, Pr un ensemble de variables et Tp l'ensemble des $\mathcal{O}$-termes sur Pr. Une $\mathcal{O}$-grammaire sur un vocabulaire Σ est constituée de l'association d'un ou plusieurs types de Tp à chaque mot de Σ. La notion de grammaire rigide (resp. k-valuée) s'applique directement à ce nouveau modèle. De même pour tout ce qui concerne les substitutions, notamment la relation $\sqsubseteq$. Une $\mathcal{O}$-règle R est une expression de la forme $A_1 \dots A_n \rightarrow A_0$, avec pour tout i A_i un $\mathcal{O}$-terme sur un ensemble fixé de variables $Var(R)$.

Soit $\mathcal{R}$ un ensemble de $\mathcal{O}$-règles : une $\mathcal{R}$-structure est un arbre sur Σ tel que chaque nœud interne est étiqueté par l'une des règles $R \in \mathcal{R}$ et chaque nœud est étiqueté par un mot du vocabulaire Σ. Un arbre de dérivation est une $\mathcal{R}$-structure dont les nœuds sont en plus étiquetés par des types, vérifiant pour chaque nœud N étiqueté par une $\mathcal{O}$-règle $R = A_1 \dots A_n \rightarrow A_0$ et un type t_0, avec n types fils $t_1, \dots, t_n$ (dans l'ordre), qu'il existe une substitution σ sur $Var(R)$ telle que $\sigma(A_i) = t_i$ pour tout $i \geq 0$. Notons que les grammaires AB sont bien sûr un exemple de formalisme représentable dans ce système.

L'algorithme d'apprentissage proposé par KANAZAWA reprend les principes de RG, en généralisant la partie "étiquetage" de la façon suivante : pour chaque structure T en entrée,

1. associer une variable distincte v_N à chaque nœud N de T et le type spécial s à la racine ;
2. pour chaque nœud N de T étiqueté par une règle $A_1 \dots A_n \rightarrow A_0$, soit τ_N un renommage de $Var(R)$ (l'ensemble d'arrivée de τ_N est constitué de variables "fraîches") : associer $x_{N_1} = \tau_N(A_1), \dots, x_{N_n} = \tau_N(A_n)$ aux nœuds fils de N et $x_N = \tau_N(A_0)$ à N ;
3. À chaque nœud sont donc associés deux types v_N et x_N : soit $\mathcal{A}$ la famille d'ensembles composée de toutes les paires (v_N, x_N) correspondant aux nœuds de T. Soit $\sigma = MGU(\mathcal{A})$: on calcule ainsi un type unique pour chaque nœud qui vérifie les contraintes imposées par les règles.

4. Collecter les types $\sigma(v_N)$ pour chaque nœud feuille N de T, et associer ce type au mot $w \in \Sigma$ correspondant à cette feuille dans la grammaire $GF(D)$.

Remarque : La seconde étape de l'algorithme, à savoir l'unification des types d'un même mot qui permet d'obtenir une grammaire rigide, est identique à celle utilisée dans le cas des grammaires AB. Une version généralisée de cet algorithme d'apprentissage de façon plus détaillée est présentée dans la partie 4.4.2.

Dans [60], KANAZAWA prouve la convergence de cet algorithme, pour n'importe quel ensemble de règles universelles $\mathcal{R}$. Nous verrons dans le chapitre 4 qu'on peut même encore étendre cette généralisation à des grammaires non algébriques (à condition évidemment d'utiliser des $\mathcal{R}$-structures adéquates), mais aussi à des grammaires pas nécessairement totalement lexicalisées. En revanche, il est primordial de noter que l'existence de cet algorithme n'implique rien quant à l'éventuelle élasticité finie de la classe des $\mathcal{R}$-grammaires rigides, ni à l'apprenabilité des $\mathcal{R}$-grammaires k-valuées ou apprises à partir de chaînes.

3.4.2.2 GCG et élasticité finie (COSTA FLORÊNCIO)

Finalement, pour les grammaires combinatoires générales, KANAZAWA laisse ouverte la question de savoir si tel ou tel ensemble de règles universelles permet d'obtenir l'élasticité finie de la classe de langages concernée (particulièrement parmi les règles des grammaires combinatoires générales, puisque c'était la raison de la généralisation qu'il proposait). Or on a vu dans le cas des grammaires AB que grâce à l'élasticité finie on obtenait des résultats positifs d'apprenabilité beaucoup plus intéressants. En effet, la restriction de l'apprenabilité des grammaires combinatoires générales aux grammaires rigides à partir de $\mathcal{R}$-structures est une limitation très contraignante pour les langues naturelles (comme pour les grammaires AB, voir partie 3.2.4). C'est pourquoi cette question est d'un intérêt certain pour l'apprenabilité de classes de langages utiles, dans des formalismes différents des grammaires AB. COSTA FLORÊNCIO a proposé dans [32] une approche aboutissant à des conditions suffisantes sur les règles universelles pour l'élasticité finie. Il faut avant tout souligner que cette question n'a rien de trivial, puisqu'on trouve facilement des classes de grammaires combinatoires générales rigides qui n'ont pas l'élasticité finie, comme en témoigne l'exemple ci-dessous tiré de [35] :

Exemple 3.2. Soit $\mathcal{R}$ l'ensemble formé des deux règles suivantes :

$$X/X\ X/X\ \rightarrow\ X$$
$$X\ Y/(X/X)\ \rightarrow\ Y$$

On montre que la classe des langages de chaînes des grammaires rigides utilisant ces règles n'est pas apprenable (donc n'a pas l'élasticité finie), en exhibant un point limite (voir théorème 2.1).

Soit G_i la grammaire $\{\ a : X/X\ ;\ b : Y/Y\ ;\ c : A_i\ \}$,
avec $A_0 = s$ et $A_i = (A_{i-1}/(Y/Y))/(X/X)$ si $i > 0$.

$G_* = \{\ a : X/X\ ;\ b : X/X\ ;\ c : s/(X/X)\ \}$. On peut montrer que $L(G_i) = c(a^*|b^*)^i$, donc $L(G_i) \subset L(G_{i+1}$ pour tout $i > 0$. De plus

$$\bigcup_{i>0} L(G_i) = c(a^*|b^*)^* = L(G_*),$$

le langage $L(G_*)$ est par conséquent un point limite pour cette classe.

Dans [32], COSTA FLORÊNCIO apporte des conditions suffisantes pour qu'un système de grammaires combinatoires générales utilisant un ensemble de règles $\mathcal{R}$ ait l'élasticité finie. L'approche qu'il propose est basée sur la réduction au cas des grammaires AB, à travers l'utilisation du théorème de KANAZAWA sur l'extension de l'élasticité finie entre classes de langages par une relation finiment valuée (voir proposition 2.14). Concrètement, le principe est le suivant : afin de replacer la classe de langages étudiée dans un cadre quasiment identique à celui des grammaires AB, une transformation des règles universelles $\mathcal{R}$ vers des règles de type AB est définie. Il faut donc aussi transformer les grammaires décrites dans le système de règles $\mathcal{R}$ en grammaires équivalentes dans ce nouveau système de règles. Bien sûr il est nécessaire que la grammaire résultant de cette transformation soit de taille finie, d'où l'existence de contraintes sur les règles universelles de départ (en plus des contraintes de forme liées aux grammaires AB, notamment le fait que les règles doivent être binaires). On obtient donc une classe de langages exprimée dans un système "de type AB", pour laquelle il existe une relation finiment valuée avec la classe des grammaires AB k-valuées. Cette dernière ayant l'élasticité finie, l'application du théorème de KANAZAWA (proposition 2.14) donne l'élasiticité finie de la classe de grammaires rigides exprimée dans le système intermédiaire. Comme dans le cas des grammaires AB, de l'élasticité finie de la classe des langages de structures des grammaires rigides découle celle de la classe des langages chaînes des grammaires k-valuée.

Il est toutefois judicieux de noter que la transformation des grammaires depuis le système de départ (règles $\mathcal{R}$) dans le système intermédiaire "de type AB" ne garantit nullement l'existence d'une éventuelle borne k sur le nombre de règles de la classe concernée, ce qui signifie que l'apprenabilité des grammaires k-valuées dans ce cas ne peut être interprétée que dans ce système intermédiaire. Le chapitre 6 sera consacré à montrer l'élasticité finie de classes de grammaires combinatoires générales selon de nouveaux critères sur les règles. Nous y proposerons notamment une comparaison avec l'approche de COSTA FLORÊNCIO dans la partie 6.6.

3.5 Conclusion

KANAZAWA a établi plusieurs résultats très importants pour l'apprenabilité des grammaires catégorielles dans [60] :

1. L'existence et la convergence de l'algorithme RG proposé auparavant par BUSZKOWSKI pour l'apprentissage des grammaires AB rigides à partir de FA-structures (théorème 3.1) ;
2. Le théorème permettant l'extension de l'élasticité finie d'une classe de langages à une autre, pourvu qu'il existe entre elles une relation finiment valuée (proposition 2.14) ;
3. L'élasticité finie de la classe des langages de FA-structures des grammaires AB rigides (proposition 3.11) ;
4. L'élasticité finie de la classe des langages de chaînes des grammaires AB k-valuées (proposition 3.13) ;

5. L'existence et la convergence d'un algorithme d'apprentissage de grammaires combinatoires générales rigides à partir de $\mathcal{R}$-structures, pour tout ensemble de $\mathcal{O}$-règles $\mathcal{R}$ (partie 3.4.2).

Mais si tous ces résultats sont bien des résultats positifs d'apprenabilité dans le modèle de GOLD au sens strict, ils ne sont pas pour autant à mettre tous sur un même plan. En effet, tant du point de vue des langues naturelles que du point de vue de l'inférence grammaticale, la dichotomie entre les points 1 et 4 de l'énumération ci-dessus est cruciale. Dans le premier cas, il existe un algorithme efficace mais l'expressivité de la classe de langages est très réduite et l'apprentissage nécessite des informations très précises en entrée. Tandis que dans le second cas, à l'inverse, l'expressivité est assez conséquente et l'apprentissage se contente de l'information la plus basique, mais il n'existe pas (et ne peut pas exister) d'algorithme efficace. Ce fossé entre apprenabilité théorique et apprenabilité pratique est sans aucun doute la plus grande difficulté à laquelle on se trouve confronté dans la problématique de l'apprentissage des langues naturelles : même si cela n'est pas trivial, globalement on peut atteindre des résultats théoriques d'apprenabilité relativement satisfaisants. En revanche, la mise en pratique de ces résultats pour les langues naturelles demeure un obstacle majeur. Nous proposerons dans le chapitre 7 une méthode qui répond partiellement à cet aspect du problème.

Du point de vue de la généralisation de ce type de résultats à d'autre formalismes, le point 5 constitue une ouverture très intéressante. En effet, il illustre le fait que la méthode d'apprentissage élaborée à partir de l'algorithme RG n'est pas propre aux grammaires AB, mais est potentiellement extensible à d'autres formes de grammaires. Cette méthode repose sur certaines propriétés relativement simples des grammaires combinatoires générales : schématiquement, les dérivations fonctionnent par substitutions de types sur un ensemble fixé de règles universelles, ce qui engendre des caractéristiques notables pour l'inclusion des grammaires par rapport à celle des langages. Nous verrons d'ailleurs dans le chapitre 4 que l'apprenabilité des langages de structures des GCG rigides ne se limite pas aux grammaires algébriques et lexicalisées. En revanche, de même que nous avons souligné la distinction entre l'apprenabilité des langages de FA-structures des grammaires AB rigides et celle des langages de chaînes des grammaires AB k-valuées, notons que ce résultat est limité aux langages de structures rigides. En effet, contrairement au cas des grammaires AB, l'élasticité finie des langages de structures rigides des GCG en général n'est pas vérifiée. Nous étudierons dans les chapitres 5 et 6 cette question laissée ouverte par KANAZAWA, à savoir quelles sous-classes parmi les langages des grammaires combinatoires générales ont l'élasticité finie. Ceci devrait permettre de découvrir de nouvelles classes de langages de grammaires k-valuées apprenables à partir de $\mathcal{R}$-structures, voire de chaînes (dans certaines conditions).

PARTIE II

Nouveaux résultats d'apprenabilité pour les langues naturelles

CHAPITRE 4

Grammaires combinatoires générales

LA VARIÉTÉ DES FORMALISMES GRAMMATICAUX ET LES PREMIERS RÉSULTATS OBTENUS PAR KANAZAWA NOUS INVITENT À CONSIDÉRER L'UTILISATION D'UN SYSTÈME FORMEL À LA FOIS FLEXIBLE ET POTENTIELLEMENT DOTÉ DE PROPRIÉTÉS POSITIVES POUR L'APPRENTISSAGE. CE SYSTÈME EST BASÉ SUR LES GRAMMAIRES COMBINATOIRES GÉNÉRALES, DONT KANAZAWA A MONTRÉ L'INTÉRÊT EN TERMES D'APPRENABILITÉ (DANS UN CAS RESTREINT). IL EST ESSENTIELLEMENT CARACTÉRISÉ PAR SON PARAMÉTRAGE PAR DES RÈGLES UNIVERSELLES, QUI S'APPLIQUENT À DES TYPES PAR SUBSTITUTIONS. DANS LA VERSION GÉNÉRALISÉE QUE NOUS PRÉSENTONS, CETTE CARACTÉRISTIQUE REND LE SYSTÈME APTE À REPRÉSENTER DES FORMALISMES ASSEZ VARIÉS. ON POURRA AINSI ÉTUDIER DE FAÇON RELATIVEMENT UNIFORME L'APPRENABILITÉ DE DIFFÉRENTS MODÈLES POUR LES LANGUES NATURELLES, TOUT EN RESTANT À UN NIVEAU DE REPRÉSENTATION AUSSI PROCHE QUE POSSIBLE DE CES MODÈLES.

4.1 Introduction

Les *Grammaires Combinatoires Générales (GCG)* sont introduites par KANAZAWA dans [60] (voir partie 3.4.2). Il s'agit en quelque sorte d'un méta-formalisme grammatical, dans lequel le système grammatical lui-même est paramétré par un ensemble fixé de règles universelles. Chaque classe de grammaires utilisant un système de règles $\mathcal{R}$ donné fonctionne par dérivation des types du lexique selon ces règles, à l'instar les grammaires AB avec les règles FA et BA. KANAZAWA proposait ce formalisme uniquement dans le but d'étudier l'apprenabilité des grammaires catégorielles combinatoires de STEEDMAN, en réutilisant les propriétés des grammaires AB dans ce cadre.

Mais cette extension offre des perspectives plus vastes pour représenter des formalismes grammaticaux un peu plus éloignés des grammaires catégorielles, comme nous le montrerons dans ce chapitre. Dans cet objectif nous proposons une définition étendue des grammaires combinatoires générales, d'une part en sortant du cadre des grammaires algébriques, et d'autre part en considérant également des grammaires non (totalement) lexicalisées. Cela permet d'envisager la description de formalismes variés dans ce système, donc potentiellement un plus grand choix de représentation des langues naturelles. Ce système se veut donc avant tout un moyen assez simple de représenter d'autres formalismes grammatiaux, à l'aide du paramétrage par les règles universelles.

Dans notre objectif d'appliquer aux langues naturelles l'apprentissage au sens du modèle de GOLD, l'intérêt des grammaires combinatoires générales réside bien sûr dans les premiers résultats d'apprenabilité obtenus par KANAZAWA sur ce système. Nous montrerons dans ce chapitre que son résultat sur l'apprentissage de grammaires rigides à partir de structures s'applique de manière assez directe à la version généralisée que nous proposons, puis nous verrons dans les chapitres suivants que certaines sous-classes de grammaires combinatoires générales, plus pertinentes en termes de représentation des langues naturelles, sont également apprenables.

4.2 GCG non restreintes

Dans cette partie nous définissons les principes de base des grammaires combinatoires générales. On parlera souvent dans la suite indifféremment de $\mathcal{R}$-grammaires ou de grammaires combinatoires générales pour évoquer l'ensemble des grammaires ou classes de grammaires incluses dans ce système.

Définition 4.1 (Règle universelle)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs. Étant donné un ensemble de variables $Var(R)$, une *règle universelle* R sur $\mathcal{O}$ est une expression de la forme

$$A_1 \dots A_n \rightarrow B_1 \dots B_m$$

avec $n > 0$, $m > 0$, et dans laquelleA_i ($1 \leq i \leq n$) et tout B_j ($1 \leq j \leq m$) est un $\mathcal{O}$-terme sur $Var(R)$.

Définition 4.2 (Système de $\mathcal{R}$-grammaires)**.** Un *sytème de $\mathcal{R}$-grammaires* est un tuple $\langle \mathcal{O}, \mathcal{R}, s \rangle$ dans lequel :

- $\mathcal{O}$ est un ensemble d'opérateurs ;
- $\mathcal{R}$ est un ensemble de règles universelles sur $\mathcal{O}$;
- s est un $\mathcal{O}$-terme sur l'ensemble vide : s est un type spécial qui caractérise les phrases correctes des langages du système.

Définition 4.3 ($\mathcal{R}$-grammaire)**.** Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires. Une $\mathcal{R}$-grammaire est un tuple $G = \langle \Sigma, Pr, \triangleright \rangle$ dans lequel :

- Σ est le vocabulaire ;
- Pr est un ensemble fini de variables, appelées les *types primitifs*. L'ensemble des types Tp est défini comme l'ensemble des $\mathcal{O}$-termes sur Pr ;
- $\triangleright$ est une relation binaire sur $(\Sigma \cup Tp)^* \times Tp^+$: Les éléments de cette relation sont appelés des *règles locales* (notées $a_1, \dots, a_n \triangleright b_1, \dots, b_m$, avec $n > 0$, $m > 0$, $a_i \in \Sigma \cup Tp$ pour tout i et $b_j \in Tp$ pour tout j).

Remarque : Dans le cadre classique des grammaires catégorielles[1] le type spécial s est généralement défini comme étant l'un des types primitifs. À l'inverse, nous proposons ici de tirer parti de la définition plus générale de l'ensemble des opérateurs $\mathcal{O}$. s est défini comme un type n'utilisant que ces opérateurs, ce qui suppose qu'il existe au moins un opérateur d'arité nulle. Ainsi s est toujours inclus dans l'ensemble des types Tp, mais existe indépendamment d'une grammaire particulière. Ceci se justifie d'autant plus

[1] De même d'ailleurs pour les grammaires classiques de Chomsky, dans lesquels le symbole S est l'un des non-terminaux.

dans le cadre de l'inférence grammaticale qu'on n'y dispose pas (par définition) des types utilisés dans une grammaire particulière, mais en revanche on sait que les objets fournis dans l'énumération sont des éléments corrects du langage, donc on peut leur "attribuer" le type s. De plus, cette définition élargie permet de prendre en compte de nouveaux formalismes assez intéressants (voir partie 4.3.4). Accessoirement, ce choix facilite techniquement la gestion du cas particulier du type s dans le cas de substitutions sur des grammaires, abondamment utilisées par la suite pour l'apprentissage.

Lorsque cela ne génère aucune ambiguité, on assimile une grammaire à l'ensemble de ses règles locales. Ainsi on note $|G|$ le nombre de règles d'une grammaire $G = \langle \Sigma, Pr, \triangleright \rangle$, c'est-à-dire $|G| = |\{ (\alpha, \beta) \mid \alpha \triangleright \beta \}|$. De même, si $G = \langle \Sigma, Pr, \triangleright \rangle$ et $G' = \langle \Sigma', Pr', \triangleright' \rangle$ sont deux $\mathcal{R}$-grammaires on dit que G est inclus dans G', noté $G \subseteq G'$, si toute règle $\alpha \triangleright \beta$ dans G implique $\alpha \triangleright' \beta$ dans G'. Enfin on dit que G est une *grammaire vide* si elle ne contient aucune règle locale.

Définition 4.4 (Dérivation simple)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs, $\mathcal{R}$ un ensemble de règles sur $\mathcal{O}$ et $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire. Pour toute règle $R \in \mathcal{R}$, avec $R = A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$, la relation $\rightarrow_R \subseteq Tp^n \times Tp^m$ est définie par :

$t_1, \ldots, t_n \rightarrow_R u_1, \ldots, u_m$ si et seulement s'il existe une substitution $\sigma : Var(R) \mapsto Tp$ telle que $\sigma(A_i) = t_i$ pour tout $1 \leq i \leq n$ et $\sigma(B_i) = u_i$ pour tout $1 \leq i \leq m$.

La relation $\rightarrow_G \subseteq (\Sigma \cup Tp)^* \times Tp^+$ est définie de la manière suivante : $t_1, \ldots, t_n \rightarrow_G u_1, \ldots, u_m$ si et seulement si

- $t_1, \ldots, t_n \triangleright u_1, \ldots, u_m$, *ou*
- il existe $R \in \mathcal{R}$ telle que $t_1, \ldots, t_n \rightarrow_R u_1, \ldots, u_m$.

À partir de cette définition il est possible de définir la dérivation sur les chaînes de la manière classique, à savoir : soit $\Rightarrow$ la relation définie par $\alpha\ \beta\ \gamma \Rightarrow \alpha\ \beta'\ \gamma$ si et seulement si $\beta \rightarrow_G \beta'$, avec $\alpha, \beta, \gamma \in (\Sigma \cup Tp)^*$ and $\beta' \in Tp^+$. On définit ensuite la relation $\Rightarrow^*$ comme la fermeture réflexive et transitive de $\Rightarrow$. Une chaîne $\alpha \in \Sigma^*$ est valide pour la $\mathcal{R}$-grammaire G si $\alpha \Rightarrow^* s$. Il est utile de noter que le sens de dérivation est défini depuis les mots du vocabulaire vers le type des phrases correctes s, à l'inverse des grammaires de constituants (définis partie 1.3.1).

En fait nous nous intéresserons beaucoup plus à l'aspect structurel des dérivations, c'est pourquoi nous proposons plutôt ci-dessous de baser la définition du langage d'une $\mathcal{R}$-grammaire sur la notion de *structure*. La définition en elle-même est certes plus complexe, mais cette vision permet de considérer, en plus des langages de chaines classiques, les langages de structures[2].

Définition 4.5 ($\mathcal{R}$-structure)**.** Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires et Σ un vocabulaire. Une *$\mathcal{R}$-structure* est une forme de graphe orienté acyclique représentant une dérivation, dans lequel les arcs représentent des mots du vocabulaire ou des types, et les nœuds les règles qui leurs sont appliquées.

L'ensemble des $\mathcal{R}$-structures $SL(\mathcal{R})$ est défini inductivement comme suit :

- Pour toute séquence $\langle w_1, \ldots, w_n \rangle$ telle que $w_i \in \Sigma$ pour tout i, la structure (figure 4.1) composée de la juxtaposition des n arcs étiquetés chacun par un w_i est une $\mathcal{R}$-structure.

[2]Dont l'utilité pour l'apprentissage est par ailleurs très grande, comme KANAZAWA l'a montré (voir partie 3.2), et comme on le verra dans les chapitres suivants.

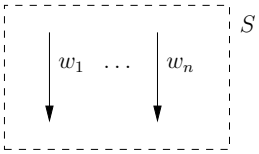


Figure 4.1 – $\mathcal{R}$-structure (mots terminaux)

Cette $\mathcal{R}$-structure a n entrées et n sorties.

- Pour toute $\mathcal{R}$-structure S ayant plus de n sorties et toute règle universelle $R \in \mathcal{R}$ avec $R = A_1, \ldots, A_n \rightarrow B_1, \ldots B_m$, la structure S' (figure 4.2) obtenue en connectant dans l'ordre les n entrées d'un nœud étiqueté R à n sorties consécutives de S est une $\mathcal{R}$-structure.

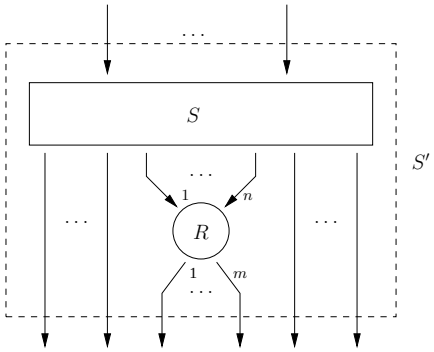


Figure 4.2 – $\mathcal{R}$-structure (application d'une règle universelle)

Les entrées de S' sont les entrées de S, et ses sorties sont (dans l'ordre) les premières sorties de S non connectées au nœud R, suivies des m sorties du nœud R, puis des dernières sorties restantes de S. Ce nœud symbolise l'application de la règle universelle R.

- Pour toute $\mathcal{R}$-structure S ayant plus de n sorties, la structure S' (figure 4.3) obtenue en connectant dans l'ordre les n entrées d'un nœud non étiqueté à n sorties consécutives de S est une $\mathcal{R}$-structure.

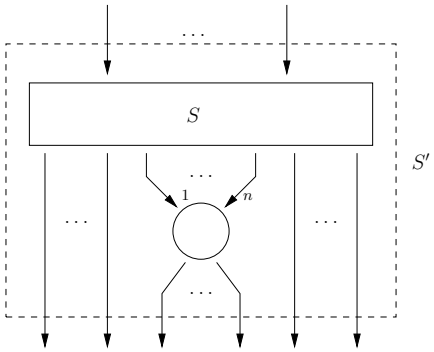


Figure 4.3 – $\mathcal{R}$-structure (application d'une règle locale)

Les entrées de S' sont les entrées de S, et ses sorties sont (dans l'ordre) les premières sorties de S

non connectées au nœud, suivies des sorties du nœud, puis des dernières sorties restantes de S. Ce nœud symbolise l'application d'une règle locale.

On voit bien en quoi notre définition étend la notion de $\mathcal{R}$-structure proposée par KANAZAWA dans [60] : les structures ne sont plus algébriques, elles ont donc potentiellement plusieurs sorties (au lieu de la seule racine d'un arbre). On abandonne également la contrainte de lexicalisation totale des grammaires, d'où la possibilité de règles locales à l'intérieur de la $\mathcal{R}$-structure (alors que l'association de types à des mots ne pouvait être faite qu'au niveau des feuilles de la structure dans la version de KANAZAWA). Bien entendu cela complexifie assez nettement la définition. Nous présenterons une version plus simple pour le cas particulier des $\mathcal{R}$-grammaires algébriques dans la définition 4.21.

La *hauteur* d'une $\mathcal{R}$-structure S, notée $h(S)$, est la longueur du plus long chemin entre une entrée et une sortie de cette structure (plus précisément, le nombre de nœuds traversés).

Définition 4.6 (Produit d'une $\mathcal{R}$-structure)**.** Le *produit* d'une $\mathcal{R}$-structure S, noté $prod(S)$, est la séquence de mots qui compose les entrées de la structure. Par définition, il ne peut s'agir que de mots du vocabulaire Σ. De manière formelle :

- si S est composée seulement de n arcs étiquetés respectivement w_1 à w_n, alors $prod(S) = \langle w_1, \ldots, w_n \rangle$;
- si S est l'application d'une règle (universelle ou locale) sur une $\mathcal{R}$-structure S', alors $prod(S) = prod(S')$.

Définition 4.7 (Structure de dérivation)**.** Soit $\mathcal{R}$ un ensemble de règles sur $\mathcal{O}$, S une $\mathcal{R}$-structure et G une $\mathcal{R}$-grammaire.

Un *étiquetage* de S est une fonction f de A vers Tp, où A est l'ensemble des arcs de S qui ne sont pas des entrées.

Soit $P = f[S]$ le résultat de l'étiquetage f appliqué à S, c'est-à-dire la structure correspondant à S dont tous les arcs (sauf les entrées) sont étiquetés par un type. Soit N un nœud de P, avec $\langle t_1, \ldots, t_n \rangle$ et $\langle u_1, \ldots, u_m \rangle$ les séquences étiquetant respectivement les entrées et les sorties de N (pour tout i, $t_i \in \Sigma \cup Tp$ et $u_i \in Tp$). L'étiquetage f est correct pour N si et seulement si

- si N est un nœud non étiqueté, alors $t_1 \ldots t_n \triangleright u_1 \ldots u_m$.
- si N est étiqueté par la règle universelle R, alors $t_1 \ldots t_n \rightarrow_R u_1 \ldots u_m$.

P est une *structure de dérivation* pour G si l'étiquetage f est correct pour tout nœud N dans P.

Une structure de dérivation P est une *instance* de S s'il existe un étiquetage f tel que $P = f[S]$.

La *sortie* d'une structure de dérivation est la séquence formée (dans l'ordre) des types et/ou mots étiquetant les arcs sortants de la structure.

Cette définition correspond à la notion d'arbre de dérivation pour les grammaires algébriques. On peut donc voir une telle structure comme une représentation graphique d'une dérivation. L'avantage d'une telle structure, comme pour un arbre de dérivation classique, est de fournir une représentation synthétique d'une dérivation. En effet, contrairement aux dérivations linéaires avec la relation $\Rightarrow$, une

structure de dérivation ne laisse pas apparaître les ambiguités dues aux différents ordres possibles de la dérivation.

Définition 4.8 (Langages d'une grammaire)**.** Soit $\mathcal{R}$ un ensemble de règles sur $\mathcal{O}$ et $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire.

- Une $\mathcal{R}$-structure $S \in SL(\mathcal{R})$ appartient au *langage de structures* de G, noté $S \in SL(G)$, si et seulement s'il existe une structure de dérivation P pour G telle que P est une instance de S et la sortie de P est égale à la séquence $\langle s \rangle$.
- Une chaîne $\alpha = w_1 \dots w_n \in \Sigma^*$ appartient au *langage de chaînes* de G, noté $\alpha \in L(G)$ si et seulement s'il existe une $\mathcal{R}$-structure $S \in SL(G)$ telle que $\langle w_1, \dots, w_n \rangle = prod(S)$.

Une structure de dérivation dont la sortie est $\langle s \rangle$ sera dite *complète*. Par opposition on parlera parfois de *structure de dérivation partielle* pour souligner que la structure en question n'est pas nécessairement complète.

On peut aisément vérifier que le langage de chaînes ainsi défini est identique à celui défini à l'aide de la relation $\Rightarrow^*$, puisque dans les deux cas chaque étape de dérivation dépend de la relation $\rightarrow_G$. De même, on peut voir que SL et L sont monotones, c'est-à-dire que si $G \subseteq G'$ alors $SL(G) \subseteq SL(G')$ et $L(G) \subseteq L(G')$ (voir proposition 4.11).

On voit dans toutes les définitions ci-dessus que l'ensemble $\mathcal{O}$ des opérateurs constitue un paramètre de l'ensemble de règles $\mathcal{R}$, qui est lui-même le paramètre de la classe des $\mathcal{R}$-grammaires. Le principe de ce formalisme est de permettre, à travers chaque système de $\mathcal{R}$-grammaires $\langle \mathcal{O}, \mathcal{R}, s \rangle$, la représentation d'une classe de grammaires/langages, dépendante de l'ensemble des règles universelles. Les grammaires combinatoires générales ne peuvent donc pas être considérées comme une classe de langages, mais plutôt comme un *modèle de représentation* des classes de langages. L'instanciation d'un tel modèle en une classe de grammaires est réalisée par la définition du système de $\mathcal{R}$-grammaires mais aussi (éventuellement) par l'ajout de contraintes supplémentaires sur les règles locales : on parle alors de *familles de $\mathcal{R}$-grammaires*. Étant donné un système de $\mathcal{R}$-grammaires $\langle \mathcal{O}, \mathcal{R}, s \rangle$, la famille la plus large définie dans ce système (c'est-à-dire celle ne comportant aucune restriction autre que celles de la définition 4.3) est appelée *famille des $\mathcal{R}$-grammaires non restreintes*.

Proposition 4.1. *Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires. Pour toute grammaire G dans ce système et toute $\mathcal{R}$-structure S, le problème de l'appartenance au langage de structures $S \in SL(G)$ est décidable*[3].

Démonstration. Nous ne donnons pas ici la démonstration en totalité car elle requiert beaucoup de détails. En revanche nous verrons dans la partie 4.4 qu'il est possible de construire une instance générique $IG(S)$ de la $\mathcal{R}$-structure S ayant la propriété suivante : si P est une instance de S alors $IG(S)$ existe et il existe une substitution σ telle que $\sigma[IG(S)] = P$ (voir proposition 4.18). Comme $S \in SL(G)$ si et seulement s'il existe une instance P de S pour G, il suffit de déterminer si une telle substitution σ existe.

[3] En d'autres termes, le langage est récursif.

Soit P une structure de dérivation constituée de l'étiquetage de chaque nœud de règle locale de S par une des règles locales de G. Si un arc est étiqueté par deux types différents, il y a évidemment échec. Si un arc n'est étiqueté par aucun type, l'étiqueter par une variable fraîche. Le nombre de telles structures P possibles est borné car $|G|$ est fini, et le nombre de nœuds de S est fini. Soit $\mathcal{A}_P$ la famille d'ensembles de types composée de toutes les paires $(t_P^E, t_{IG(S)}^E)$ pour tout arc E de S, avec t_P^E (resp. $t_{IG(S)}^E$) le type associé à E dans P (resp. dans $IG(S)$) : il existe une telle structure de dérivation P telle que l'unfieur le plus général (MGU) de $\mathcal{A}_P$ existe si et seulement si $S \in SL(G)$ (grâce à la proposition 4.18). □

Proposition 4.2. *Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires. Pour toute grammaire G dans ce système et toute chaîne x, le problème de l'appartenance au langage de chaînes $x \in L(G)$ est semi-décidable*[4].

Démonstration. Le nombre de $\mathcal{R}$-structures de hauteur inférieure ou égale à n dont le produit est x est borné, car la taille des règles (universelles ou locales) est borné. Ainsi il est possible d'énumérer l'ensemble des $\mathcal{R}$-structures possibles, par ordre de hauteur maximale croissante. Pour chaque $\mathcal{R}$-structure, le problème de l'appartenance au langage de structures est décidable, donc s'il existe une $\mathcal{R}$-structure (dont le produit est x) appartenant au langage de structures, celle-ci sera atteinte au bout d'un temps fini. □

Proposition 4.3. *Il existe une famille de $\mathcal{R}$-grammaires équivalente à la classe des langages récursivement énumérables.*

Démonstration. On prend simplement $\mathcal{O}_{re} = \mathcal{R}_{re} = \emptyset$: la définition d'une $\mathcal{R}_{re}$-grammaire non restreinte est ainsi exactement identique à celle d'une grammaire de type 0, donc les deux classes de langages sont clairement équivalentes. □

On voit donc que notre définition très large des grammaires combinatoires générales comprend l'ensemble des langages récursivement énumérables. L'expressivité du modèle étant très vaste, il est possible de représenter toutes sortes de classes de langages. En fait la principale limitation de ce modèle se situe plutôt dans sa forme : pour qu'un formalisme grammatical puisse être utilisé dans ce cadre, il doit être possible de l'exprimer sous forme de règles de réécriture, ce qui n'est pas toujours le cas de manière directe. Nous présentons dans la prochaine partie certaines sous-classes de GCG basées sur des restrictions qui donnent à cette sous-classe certaines propriétés intéressantes. Avant cela, nous présentons encore quelques définitions qui seront utiles par la suite.

Définition 4.9 ($Var(G)$)**.** L'ensemble des variables d'une grammaires G noté $Var(G)$ est défini comme l'ensemble des types primitifs qui apparaissent au moins une fois dans l'une des règles locales de la grammaire.

Définition 4.10 (Taille d'une grammaire)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire.

La taille d'une règle locale $r = a_1 \dots a_n \triangleright b_1 \dots b_m$, notée $\|r\|$, est définie par

$$\|r\| = \sum_{i=1}^{n} \|a_i\| + \sum_{i=1}^{m} \|b_i\|,$$

[4] En d'autres termes, le langage est récursivement énumérable.

avec $\|a_i\| = 1$ si $a_i \in \Sigma$.

La taille d'une grammaire G, notée $\|G\|$, est définie par

$$\|G\| = \sum_{r \in G} \|r\|.$$

Définition 4.11 (Types réalisables, types faisables)**.** Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires, et G une $\mathcal{R}$-grammaire dans ce système.

L'ensemble $RT(G)$ des *types réalisables* pour G est défini comme le plus petit ensemble tel que

- Pour toute règle locale $a_1 \dots a_n \triangleright b_1 \dots b_m$ dans G telle que $a_i \in \Sigma$ pour tout i ($1 \leq i \leq n$), $\{b_1, \dots, b_m\} \subseteq RT(G)$;
- Quels que soient $t_1, \dots, t_n$ tels que $\{t_1, \dots, t_n\} \subseteq RT(G)$, si $t_1 \dots t_n \rightarrow_G u_1 \dots u_m$ alors $\{u_1, \dots, u_m\} \subseteq RT(G)$.

L'ensemble $FT(G)$ des *types faisables* pour G est défini par :

Quel que soit $t \in RT(G)$, $t \in FT(G)$ si et seulement s'il existe une séquence de types $\langle t_1, \dots, t_n \rangle$ et un entier i, $1 \leq i \leq n$, tels que $\{t_1, \dots, t_n\} \subseteq RT(G)$, $t_i = t$ et $t_1 \dots t_n \Rightarrow^* s$.

En termes moins formels, un type n'est réalisable que s'il peut apparaître dans une structure de dérivation partielle, et il est de plus faisable s'il peut apparaître dans une structure de dérivation complète.

4.2.1 Propriétés et sous-classes de grammaires combinatoires générales

Définition 4.12 (Substitution sur une grammaire)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire. L'application d'une substitution $\sigma : Pr \mapsto Tp'$ à une règle (locale) $r = a_1, \dots, a_n \triangleright b_1, \dots, b_m$ de G est la règle $\sigma(r) = \sigma(a_1), \dots, \sigma(a_n) \triangleright' \sigma(b_1), \dots, \sigma(b_m)$, avec $\sigma(a_i) = a_i$ si $a_i \in \Sigma$[5].

L'application d'une substitution $\sigma : Pr \mapsto Tp'$ à une grammaire $G = \langle \Sigma, Pr, \triangleright \rangle$, notée $\sigma[G]$, est la grammaire $G' = \langle \Sigma', Pr', \triangleright' \rangle$ définie par $\triangleright' = \{ \sigma(r) \mid r \in \triangleright \}$.

Intuitivement, $\sigma[G]$ est la grammaire G dans laquelle tout type t est remplacé par $\sigma(t)$ dans toutes les règles locales.

Exemple 4.1 (KANAZAWA)**.** Soient G_1 et G_2 les deux grammaires suivantes :

$$G_1 = \{\ fast \triangleright y\backslash(x\backslash s)\ ;\ John \triangleright x\ ;\ walks \triangleright x\backslash s, y\ \}, \text{ et}$$

$$G_2 = \{\ fast \triangleright (x\backslash s)\backslash(x\backslash s)\ ;\ John \triangleright x\ ;\ walks \triangleright x\backslash s\ \}.$$

Il existe une substution $\sigma = \{y \mapsto x\backslash s\}$ telle que $G_2 = \sigma[G_1]$.

[5] En d'autres termes les éléments du vocabulaires sont considérés comme des constantes (ou opérateurs d'arité nulle) pour les substitutions.

Définition 4.13 (Grammaires équivalentes)**.** Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires, et $G = \langle \Sigma, Pr, \triangleright \rangle$ et $G' = \langle \Sigma', Pr', \triangleright' \rangle$ deux $\mathcal{R}$-grammaires. G et G' sont équivalentes, noté $G \equiv G'$, si et seulement s'il existe deux substitutions σ_1 et σ_2 telles que $\sigma_1[G] = G'$ et $\sigma_2[G'] = G$.

L'existence de deux substitutions réciproques l'une de l'autre entre les grammaires est équivalente à l'existence d'une substitution bijective d'une grammaire à l'autre, autrement dit à un renommage de variables (voir définition 1.7).

Définition 4.14 (Règle strictement décroissante[6])**.** Une règle universelle $R = A_1 \dots A_n \rightarrow B_1 \dots B_m$ est strictement décroissante si et seulement si $n > m$.

Définition 4.15. Soit k un entier positif : une grammaire G est *k-bornée* si et seulement si $|G| \leq k$.

4.2.1.1 GCG lexicalisées

Définition 4.16 (Grammaire lexicalisée)**.** Une grammaire $G = \langle \Sigma, Pr, \triangleright \rangle$ est *lexicalisée* si toute règle locale dans G est de la forme $w \triangleright t$, avec $w \in \Sigma$ et $t \in Tp$.

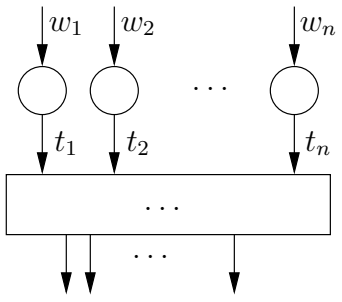
L'ensemble de types $\{\, t \in Tp \mid$ il existe $w \in \Sigma$ tel que $w \triangleright t \,\}$ est noté $Lex(G)$.

Étant donné un système de $\mathcal{R}$-grammaires, la famille des $\mathcal{R}$-grammaires lexicalisées est constituée du sous-ensemble des $\mathcal{R}$-grammaires non restreintes constitué uniquement des grammaires lexicalisées.

Définition 4.17 (Sous-types d'une grammaire lexicalisée)**.** L'ensemble des sous-types d'une grammaire lexicalisée $G = \langle \Sigma, Pr, \triangleright \rangle$, noté $STLex(G)$, est l'ensemble

$$STLex(G) = \{\, t \mid \text{il existe un type } t' \in Lex(G) \text{ tel que } t \text{ est un sous-type de } t' \,\}$$

Si G est une $\mathcal{R}$-grammaire lexicalisée, alors toute structure de dérivation pour G est de la forme



[6] La définition que nous proposons des *règles strictement décroissantes* correspond exactement à celle des *grammaires contextuelles croissantes (growing context-sensitive grammars)* de DAHLAUS et WARMUTH [36]. Dans le cas des grammaires de constituants contextuelles (étudié par DAHLAUS et WARMUTH), cette restriction a la particularité de permettre de résoudre le problème de l'appartenance en temps polynomial [36]. Cependant nous utilisons cette restriction dans un contexte différent : d'une part par l'orientation des règles (inversée), et d'autre part par la présence de termes (et non de symboles atomiques) dans les règles. C'est pourquoi nous utilisons une terminologie différente, car les propriétés des grammaires contextuelles croissantes ne sont pas nécessairement applicables aux GCG.

avec $w_i \triangleright t_i$ pour tout i. Dans ce cas, la séquence $\langle t_1, t_2, \ldots, t_n \rangle$ est appelée l'*entrée* de la structure de dérivation. De plus, le reste de la structure ne peut être composé que de nœuds correspondant à des applications de règles universelles. Les $\mathcal{R}$-structures appartenant au langage de structures d'une grammaire lexicalisée ont donc aussi cette forme particulière, car il existe toujours une structure de dérivation qui est une instance d'une telle $\mathcal{R}$-structure, par définition.

On reconnaît bien sûr dans cette lexicalisation totale l'une des caractéritiques des grammaires catégorielles. On verra dans les chapitres suivants que la lexicalisation facilite grandement l'apprentissage, et donc "augmente les chances" qu'une classe de grammaires soit apprenable. En effet, le fait que les règles soient ancrées au niveau des mots du vocabulaire, toujours visibles dans les exemples fournis en entrée de l'algorithme d'apprentissage, guide celui-ci dans le choix des règles à "deviner".

Définition 4.18 (Grammaire k-valuée)**.** Une grammaire lexicalisée $G = \langle \Sigma, Pr, \triangleright \rangle$ est k-valuée si pour tout mot $w \in \Sigma : |\{ t \in Tp \mid w \triangleright t \}| \leq k$.

Proposition 4.4. *Soit $\mathcal{G}$ une classe de grammaires sur un vocabulaire Σ fixé :*

i. Pour tout $k \geq 0$, il existe un k' tel que $\{ G \in \mathcal{G} \mid G$ est k-valuée $\} \subseteq \{ G \in \mathcal{G} \mid G$ est k'-bornée $\}$.

ii. Pour tout $k \geq 0$, il existe un k' tel que $\{ G \in \mathcal{G} \mid G$ est k-bornée $\} \subseteq \{ G \in \mathcal{G} \mid G$ est k'-valuée $\}$.

Démonstration. (i) On prend $k' = k \times |\Sigma|$: toute grammaire k-valuée ne peut avoir plus de k' règles.
(ii) On prend $k' = k$: toute grammaire ayant au plus k règles est nécessairement k-valuée. □

4.2.1.2 GCG contextuelles

Définition 4.19 (Famille de $\mathcal{R}$-grammaires contextuelles)**.** $\langle \mathcal{O}, \mathcal{R}, s \rangle$ est un système de $\mathcal{R}$-grammaires *contextuel* si pour toute règle $R = A_1 \ldots A_n \rightarrow B_1 \ldots B_m$ de $\mathcal{R}$

$$\sum_{i=1}^{n} \|A_i\| \geq \sum_{i=1}^{m} \|B_i\| \quad \text{et} \quad \forall v \in Var(R), \sum_{i=1}^{n} \#_v(A_i) \geq \sum_{i=1}^{m} \#_v(B_i)$$

Si de plus $n \geq m$, alors $\langle \mathcal{O}, \mathcal{R}, s \rangle$ est dit *fortement contextuel*.

Si $\langle \mathcal{O}, \mathcal{R}, s \rangle$ est contextuel, la *famille des $\mathcal{R}$-grammaires contextuelles* est l'ensemble des grammaires dont toutes les règles locales $a_1 \ldots a_n \triangleright b, \ldots b_m$ vérifient

$$\sum_{i=1}^{n} \|a_i\| \geq \sum_{i=1}^{m} \|b_i\| \quad \text{ou} \quad \text{il existe un } i(1 \leq i \leq n) \text{ tel que } a_i \in \Sigma$$

.

Si de plus $n \geq m$, alors la grammaire appartient aussi à la sous-famille des *$\mathcal{R}$-grammaires fortement contextuelles*.

Proposition 4.5. *Soit R une règle universelle contextuelle. Si $t_1 \ldots t_n \rightarrow_R u_1 \ldots u_m$ alors*

$$\sum_{i=1}^{n} \|t_i\| \geq \sum_{i=1}^{m} \|u_i\|$$

Démonstration. Soit $R = A_1 \dots A_n \to B_1 \dots B_m$, avec $\mathcal{V} = Var(R)$. $t_1 \dots t_n \to_R u_1 \dots u_m$, donc il existe une substitution σ telle que $\sigma(A_i) = t_i$ et $\sigma(B_j) = u_j$ pour tout $1 \leq i \leq n$ et tout $1 \leq j \leq m$.

$$\begin{aligned}
\sum_{i=1}^{n} \|t_i\| &= \sum_{i=1}^{n} \|\sigma(A_i)\| \\
&= \sum_{i=1}^{n} (\|A_i\| + \sum_{v \in \mathcal{V}} (\#_v(A_i) \times (\|\sigma(v)\| - 1))) \\
&= \sum_{i=1}^{n} \|A_i\| + (\sum_{i=1}^{n} \sum_{v \in \mathcal{V}} (\#_v(A_i)) \times (\|\sigma(v)\| - 1)) \\
&\geq \sum_{i=1}^{m} \|B_i\| + (\sum_{i=1}^{m} \sum_{v \in \mathcal{V}} (\#_v(B_i)) \times (\|\sigma(v)\| - 1)) \\
&\geq \sum_{i=1}^{m} \|u_i\|
\end{aligned}$$

□

Proposition 4.6. *Pour toute grammaire G appartenant à une famille de $\mathcal{R}$-grammaires contextuelles et toute chaîne x, le problème de l'appartenance au langage de chaînes $x \in L(G)$ est décidable.*

Démonstration. Soit $x = w_1 \dots w_n$. On montre par induction sur i que dans toute dérivation complète de x $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \dots \Rightarrow \alpha_m$, avec $\alpha_0 = x$ et $\alpha_m = \langle s \rangle$, la longueur de chaque α_i est bornée. Soit f la fonction définie par $f(y) = 0$ si $y \in \Sigma$ et $f(y) = \|y\|$ si $y \in Tp$, et g la fonction définie par $g(y) = 1$ si $y \in \Sigma$ et $g(y) = 0$ sinon. Pour toute séquence $\alpha = a_1 \dots a_j \subseteq (\Sigma \cup Tp)^*$, on définit

$$t(\alpha) = \sum_{k=1}^{j} f(a_k) \text{ et } v(\alpha) = \sum_{k=1}^{j} g(a_k).$$

Soit $M = max(\{\, t(\beta) \mid \alpha \triangleright \beta$ est une règle locale dans $G \,\})$. Montrons par induction sur i que pour tout i ($0 \leq i \leq m$) on a $t(\alpha_i) \leq (n - v(\alpha_i)) \times M$:

$i = 0$. $\alpha_0 = w_1 \dots w_n$ avec $w_j \in \Sigma$ pour tout $j \leq n$, donc $t(\alpha_0) = 0$ et $v(\alpha_0) = n$.

$i > 0$. On suppose que $t(\alpha_{i-1}) \leq (n - v(\alpha_{i-1})) \times M$. $\alpha_{i-1} \Rightarrow \alpha_i$ donc il existe $\beta_1\beta_2\beta_3, \gamma$ tels que $\alpha_{i-1} = \beta_1\beta_2\beta_3$, $\alpha_i = \beta_1\gamma\beta_3$ et $\beta_2 \to_G \gamma$.

- Si $v(\beta_2) = 0$: la dérivation utilise une règle contextuelle, donc (d'après la proposition 4.5 s'il s'agit d'une règle universelle ou par définition si c'est une règle locale) $t(\beta_2) \geq t(\gamma)$, et comme il ne peut y avoir aucun $w \in \Sigma$ dans γ on a $v(\gamma) = v(\beta_2)$. Ainsi $t(\alpha_i) = t(\beta_1) + t(\gamma) + t(\beta_3) \leq t(\beta_1) + t(\beta_2) + t(\beta_3) = t(\alpha_{i-1}) \leq (n - v(\alpha_{i-1})) \times M = (n - v(\alpha_i)) \times M$.
- Si $v(\beta_2) > 0$, la dérivation passe par une règle locale $\beta_2 \triangleright \gamma$. Par définition de M, $t(\gamma) \leq M$, donc $t(\gamma) \leq t(\beta_2) + M$. Par conséquent $t(\alpha_i) = t(\beta_1) + t(\gamma) + t(\beta_3) \leq t(\beta_1) + t(\beta_2) + t(\beta_3) + M = t(\alpha_{i-1}) + M$, donc $t(\alpha_i) \leq (n - v(\alpha_{i-1})) \times M + M$ par hypothèse d'induction. Comme $v(\beta_2) > 0$ et $v(\gamma) = 0$ (par définition), $v(\alpha_i) = v(\beta_1) + v(\gamma) + v(\beta_3) \leq v(\beta_1) + v(\gamma) + v(\beta_3) - 1 = v(\alpha_{i-1}) - 1$. $v(\alpha_i) \leq v(\alpha_{i-1}) - 1$ implique $(n - v(\alpha_i)) \times M \geq (n - v(\alpha_{i-1})) \times M + M \geq t(\alpha_i)$.

Ainsi pour tout i $t(\alpha_i) \leq (n - v(\alpha_i)) \times M \leq n \times M$. Par conséquent il suffit pour trouver une dérivation éventuelle de x de tester toutes les combinaisons de séquences de types ayant une taille inférieure ou égale à $|x| \times M$.

□

Dans le cas particulier des $\mathcal{R}$-grammaires fortement contextuelles, une transformation simple permet de convertir une $\mathcal{R}$-grammaire en grammaire de constituants contextuelle. Cette transformation est l'objet de la proposition 4.7 ci-dessous.

Proposition 4.7. *Pour toute grammaire G appartenant à une famille de $\mathcal{R}$-grammaires fortement contextuelles, il existe une grammaire contextuelle équivalente.*

Démonstration. Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire fortement contextuelle. Soit E l'ensemble des types apparaissant en partie droite d'une règle locale de G. Soit $f_{\mathcal{R}}$ la fonction définie sur un ensemble fini de types T par

$$f_{\mathcal{R}}(T) = T \cup \{\, u_i \in Tp \mid \exists t_1, \ldots, t_n \in T \text{ et } \exists R \in \mathcal{R} \text{ tels que } t_1 \ldots t_n \rightarrow_R u_1 \ldots u_m \text{ et } 1 \leq i \leq m \,\}$$

D'après la proposition 4.5, tout type dans $f(T)$ est de taille inférieure ou égale à $n \times m$, avec n le nombre maximum de termes dans la partie gauche d'une règle de $\mathcal{R}$ et m la taille maximale d'un type de T. Par conséquent l'ensemble $f^*(E)$ constitué de la fermeture réflexive et transitive de E par f est fini. On peut donc construire une grammaire (finie) de type 1 de la façon suivante : soit g une bijection de $f^*(E)$ dans l'ensemble N des non-terminaux de la grammaire de type 1. Pour tous types $t_1, \ldots, t_n, u_1, \ldots, u_m \in f^*(E)$ tels que $t_1 \ldots t_n \rightarrow_G u_1 \ldots u_m$, créer la règle $g(u_1) \ldots g(u_m) \rightarrow g(t_1) \ldots g(t_n)$. $m \leq n$ car G est fortement contextuelle, donc la grammaire construite est bien de type 1. □

On peut voir cette transformation comme une généralisation de la transformation proposée dans [12] pour transformer une grammaire AB en grammaire algébrique (voir partie 1.3.2.1).

4.2.1.3 GCG algébriques

Définition 4.20 (Famille de $\mathcal{R}$-grammaires algébriques)**.** $\langle \mathcal{O}, \mathcal{R}, s \rangle$ est un système de $\mathcal{R}$-grammaires *algébriques* si toute règle $R \in \mathcal{R}$ est de la forme $A_1 \ldots A_n \rightarrow A_0$, avec

$$\sum_{i=1}^{n} \|A_i\| \geq \|A_0\| \quad \text{et} \quad \forall v \in Var(R), \sum_{i=1}^{n} \#_v(A_i) \geq \#_v(A_0)$$

Si $\langle \mathcal{O}, \mathcal{R}, s \rangle$ est algébrique, la *famille des $\mathcal{R}$-grammaires algébriques* est l'ensemble des grammaires dont toutes les règles locales sont de la forme $a_1 \ldots a_n \triangleright a_0$ et vérifient

$$\sum_{i=1}^{n} \|a_i\| \geq \|a_0\| \quad \text{ou} \quad \text{il existe un } i \ (1 \leq i \leq n) \text{ tel que } a_i \in \Sigma.$$

Remarque : Comme on peut s'y attendre, une famille de $\mathcal{R}$-grammaires algébriques est toujours aussi une famille (fortement) contextuelle. La forme des règles d'une $\mathcal{R}$-grammaire algébrique autorise une définition plus simple des $\mathcal{R}$-structures dans ce cas. Cette définition (ci-dessous) est équivalente à la précédente, sauf qu'il s'agit désormais d'un arbre classique (c'est-à-dire que seuls les nœuds sont étiquetés. En particulier les nœuds feuilles de l'arbre représentent les mots de la chaîne). On retrouve ainsi une définition semblable à celle utilisée par KANAZAWA dans [60] (à la lexicalisation près).

Définition 4.21 ($\mathcal{R}$-structure algébrique)**.** Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires algébriques et Σ un vocabulaire. L'ensemble des $\mathcal{R}$-structures $SL(\mathcal{R})$ est le plus petit ensemble tel que

- Tout mot $w \in \Sigma$ appartient à $SL(\mathcal{R})$,
- Pour toute règle $R \in \mathcal{R}$ telle que $R = A_1 \dots A_n \rightarrow A_0$, si $T_1, \dots, T_n \in SL(\mathcal{R})$ alors $[R](T_1, \dots, T_n) \in SL(\mathcal{R})$.
- Pour toute séquence de $\mathcal{R}$-structures $T_1, \dots, T_n$, $[\,](T_1, \dots, T_n) \in SL(\mathcal{R})$ ($[\,]$ est un nœud non étiqueté correspondant à l'application d'une règle locale).

Le produit $prod(S)$ d'une $\mathcal{R}$-structure algébrique est défini comme la séquence de mots apparaissant sur les feuilles de l'arbre. De manière similaire on peut utiliser la définition suivante pour décrire une structure de dérivation algébrique (on parle alors d'*arbre de dérivation*) :

Définition 4.22 (Structure de dérivation algébrique)**.** Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires algébriques, S une $\mathcal{R}$-structure et G une $\mathcal{R}$-grammaire algébrique. P est une instance de S pour une grammaire $G = \langle \Sigma, Pr, \triangleright \rangle$ si

- $S = w \in \Sigma$ et $P = w$. $racine(P) = w$.
- $S = [R](S_1, \dots, S_n)$ et $P = [R](t, P_1, \dots, P_n)$ sont tels que P_i est une instance de S_i pour tout i, et $t_1 \dots t_n \rightarrow_R t$ avec $t_i = racine(P_i)$ pour tout i. $racine(P) = t$.
- $S = [\,](S_1, \dots, S_n)$ et $P = [\,](t, P_1, \dots, P_n)$ sont tels que P_i est une instance de S_i pour tout i, et $t_1 \dots t_n \triangleright t$ avec $t_i = racine(P_i)$ pour tout i. $racine(P) = t$.

Une $\mathcal{R}$-structure algébrique $S \in SL(\mathcal{R})$ appartient au *langage de structures* de G, noté $S \in SL(G)$, si et seulement s'il existe une structure de dérivation P pour G telle que P est une instance de S et $racine(P) = s$.

Proposition 4.8. *Pour toute grammaire G appartenant à une famille de $\mathcal{R}$-grammaires algébriques, il existe une grammaire algébrique équivalente.*

Démonstration. Il s'agit d'un cas particulier simple de la proposition 4.7. Comme pour le cas contextuel, l'ensemble des types utilisables dans une dérivation est fini. Chaque règle n'a qu'un seul type à droite, donc les règles construites pour la grammaire cible vérifient les contraintes des grammaires algébriques. □

4.2.1.4 Opérateurs à arguments

Le fait que les opérateurs des grammaires AB se décomposent chacun en une partie dite *principale* et une partie dite *argument* constitue un des points clé des possibilités en termes d'apprentissage sur ces grammaires. Dans le type A/B, le sous-type A est la partie principale et le sous-type B est la partie argument. Ces termes traduisent la sémantique sous-jacente aux règles universelles des grammaires AB : lorsque la règle $A/B\ B \rightarrow A$ est utilisée (il s'agit de l'unique règle utilisable pour décomposer un type de la forme A/B), on constate que le type B (l'argument) est consommé pour produire le type A (la partie principale). Ceci entraîne un certain nombre de propriétés remarquables, dont certaines sont nécessaires pour l'apprenabilité, comme on peut le voir dans plusieurs démonstrations de KANAZAWA [60].

C'est la raison pour laquelle nous introduisons, dans la présente généralisation aux grammaires combinatoires générales, la distinction entre position *principale* et position *argument* des opérateurs. Lorsque cette distinction sera nécessaire on parlera d'un opérateur f *muni d'une fonction* arg_f, cette fonction étant définie de la façon suivante :

Définition 4.23 (Opérateur muni d'une fonction arg)**.** Soit n l'arité de l'opérateur f. La fonction $arg_f : \{1, \ldots, n\} \mapsto \{0, 1\}$ est définie pour tout i, $1 \leq i \leq n$, par

- $arg_f(i) = 1$ si la i^{e} position de l'opérateur f est une position argument ;
- $arg_f(i) = 0$ sinon.

Par exemple pour suivre la sémantique habituelle des opérateurs $/$ et$\backslash$des grammaires AB, on définira

- $arg_/$ par $arg_/(1) = 0$ et $arg_/(2) = 1$;
- $arg_\backslash$ par $arg_\backslash(1) = 1$ et $arg_\backslash(2) = 0$.

En effet dans le type A/B c'est l'opérande de droite B, correspondant à la seconde position de l'opérateur $/$, qui est l'argument.

Il faut noter qu'aucune sémantique particulière n'est associée a priori aux positions arguments. En revanche nous utiliserons cette distinction dans les chapitres 5 et 6, en imposant des contraintes syntaxiques particulières sur certaines positions, de manière à obtenir des propriétés favorables pour l'apprenabilité des classes grammaires respectant ces contraintes. Comme on peut le voir nous n'avons pas défini dans cette partie de telles contraintes, car nous étudierons les effets de deux types de contraintes distincts, aboutissant à deux résultats d'apprenabilité différents.

4.3 Quelques formalismes représentables

Le système des grammaires combinatoires générales que nous proposons n'a pas d'autre intérêt que sa capacité à représenter, dans un cadre uniformisé, des formalismes grammaticaux variés. Dans sa version des GCG, KANAZAWA ne considérait que la possibilité de représenter les grammaires catégorielles combinatoires de STEEDMAN. Nous montrons dans cette partie que ce système offre une souplesse suffisante pour gérer des formalismes bien plus éloignés des grammaires catégorielles, tout en restant évidemment adapté à ces dernières. Les quelques exemples donnés ici ne sont bien entendu pas exhaustifs : tout système exprimable en termes de règles universelles de réécriture est suceptible d'entrer dans ce cadre.

4.3.1 Grammaires de constituants

Les grammaires de constituants sont le premier formalisme grammatical auquel on peut s'intéresser (voir partie 1.3.1). Il s'agit d'un cas assez trivial du point de vue des GCG, puisque la définition des GCG non restreintes est de toute évidence plus large que celle des grammaires de constituants[7]. Il suffit en effet que l'ensemble des opérateurs et celui des règles universelles soient tous deux définis comme

[7] On notera cependant la différence d'orientation des règles locales (parties gauche et droite inversées), qui engendre également une différence d'orientation des dérivations.

l'ensemble vide pour obtenir une définition totalement équivalente à celle des grammaires de constituants (comme dans la proposition 4.3). Par ailleurs on retrouve bien sûr les trois sous-classes de la hiérarchie de CHOMSKY en appliquant les contraintes appropriées sur la forme des règles locales.

Exemple 4.2 (Grammaire contextuelle)**.** Soit G la grammaire contextuelle définie par les règles locales[8] suivantes :

$R_1 : aSBc \rightarrow S$

$R_2 : abc \rightarrow S$

$R_3 : Bc \rightarrow cB$

$R_4 : bb \rightarrow bB$

Le langage généré par cette grammaire est $L(G) = \{a^n b^n c^n | n > 0\}$. La figure 4.4 représente la structure de dérivation de la phrase "$aaabbbccc$" pour cette grammaire.

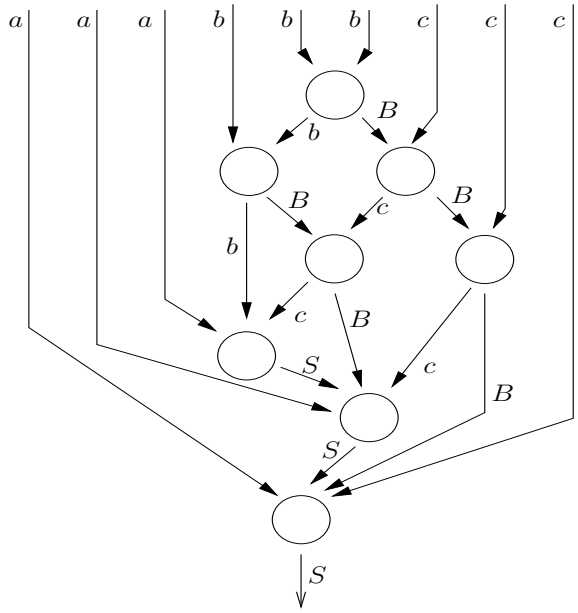


Figure 4.4 – Structure de dérivation d'une grammaire contextuelle

L'exemple 4.2 illustre la représentation classique des grammaires de constituants sous forme de GCG, sans règles universelles. Mais les langages rationnels étant aussi représentables sous forme d'expressions régulières, on peut représenter cette classe précise à l'aide des opérateurs classiques (définis dans la partie 1.3.1.2) et de règles universelles adéquates, comme dans l'exemple ci-dessous.

Exemple 4.3 (Expressions régulières)**.** L'ensemble des opérateurs est $\mathcal{O}_{re} = \{ \bullet(2) ; {}^*(1) ; \cup(2) \}$. Les règles universelles $\mathcal{R}_{re}$ sont :

[8] Remarque : cette grammaire ne vérifie pas totalement la définition d'une $\mathcal{R}$-grammaire, dans la mesure où la contrainte d'absence de mots terminaux en partie droite des règles n'est pas respectée. Cette simplification vise seulement à diminuer le nombre de règles : il est clair qu'il est possible de définir la même grammaire en respectant cette contrainte.

$$
\begin{array}{l}
A \quad B \rightarrow A \bullet B \\
\varepsilon \rightarrow A^* \\
A \quad A^* \rightarrow A^* \\
A \rightarrow A \cup B \\
A \rightarrow B \cup A
\end{array}
$$

Une grammaire dans cette famille est constituée d'une unique règle de la forme $E \triangleright s$, avec $E \in Tp_{re}$, et d'un nombre quelconque de règles de la forme $w \triangleright t$, avec $t \in Pr$. Ainsi l'expression régulière $(a^*b^*(c \cup a)) \cup c^*$ peut être représentée par la grammaire G ci-dessous :

$$G = \{\ ((A^* \bullet B^*) \bullet (C \cup A)) \cup C^* \triangleright s\ ;\ a \triangleright A\ ;\ b \triangleright B\ ;\ c \triangleright C\ \}.$$

Avec cette grammaire on peut par exemple dériver la chaîne $aaac$ de la façon suivante :
$a\ a\ a\ c \rightarrow^* A\ A\ A\ \varepsilon\ \varepsilon\ C \rightarrow A\ A\ A\ A^*\ \varepsilon\ C \rightarrow A\ A\ A\ A^*\ B^*\ C \rightarrow A\ A\ A^*\ B^*\ C$
$\rightarrow A\ A^*\ B^*\ C \rightarrow A^*\ B^*\ C \rightarrow (A^* \bullet B^*)\ C \rightarrow (A^* \bullet B^*)\ (C \cup A)$
$\rightarrow (A^* \bullet B^*) \bullet (C \cup A) \rightarrow ((A^* \bullet B^*) \bullet (C \cup A)) \cup C^* \rightarrow s$

4.3.2 Grammaires catégorielles

Remarque : toutes les familles de grammaires combinatoires générales définies ci-dessous sont algébriques et lexicalisées.

Les grammaires catégorielles classiques, ou grammaires AB, définies dans la partie 1.3.2.1, sont évidemment facilement représentables dans le système des grammaires combinatoires générales. L'ensemble des opérateurs[9] est $\mathcal{O}_{AB} = \{\ /(2)\ ;\ \backslash(2)\ ;\ s(0)\ \}$, et l'ensemble des règles universelles $\mathcal{R}_{AB}$ est constitué des deux règles FA et BA :

$$\mathcal{R}_{AB} = \left\{ \begin{array}{lll} (R_{FA}) & A/B \quad B \rightarrow A & (\text{avec } Var(R_{FA}) = \{A, B\})\ ; \\ (R_{BA}) & B \quad B\backslash A \rightarrow A & (\text{avec } Var(R_{BA}) = \{A, B\}) \end{array} \right\}$$

Il n'est pas difficile de voir que cette famille de $\mathcal{R}$-grammaires est totalement équivalente à la classe des grammaires AB. En particulier on peut noter que les langages de $\mathcal{R}$-structures de cette famille correspondent exactement aux langages de FA-structures des grammaires AB.

De la même façon, il est possible de représenter les règles des grammaires catégorielles combinatoires de STEEDMAN (voir partie 1.3.2.3) dans ce système, comme cela était déjà le cas avec la version proposée par KANAZAWA dans [60]. Les opérateurs restent identiques à ceux des grammaires AB : $\mathcal{O}_{CCG} = \mathcal{O}_{AB}$. L'ensemble de règles[10] $\mathcal{R}_{CCG}$ est défini par :

[9] On notera l'introduction du type spécial s parmi les opérateurs, conformément à la définition 4.3. Voir les explications dans la partie 4.2.

[10] Rappel : par convention, nous n'utilisons pas la notation usuelle des grammaires catégorielles combinatoires pour l'opérateur $\backslash$ (voir les explications dans la partie 1.3.2.3).

$$\mathcal{R}_{CCG} = \left\{ \begin{array}{lllll} \text{Forward Application} & X/Y & Y & \rightarrow & X & Var(R) = \{X, Y\} ; \\ \text{Backward Application} & Y & Y\backslash X & \rightarrow & X & Var(R) = \{X, Y\} ; \\ \text{Forward Composition} & X/Y & Y/Z & \rightarrow & X/Z & Var(R) = \{X, Y, Z\} ; \\ \text{Backward Composition} & Z\backslash Y & Y\backslash X & \rightarrow & Z\backslash X & Var(R) = \{X, Y, Z\} ; \\ \text{Forward Type Raising} & X & & \rightarrow & Y/(X\backslash Y) & Var(R) = \{X, Y\} ; \\ \text{Backward Type Raising} & X & & \rightarrow & (Y/X)\backslash Y & Var(R) = \{X, Y\} \end{array} \right\}$$

Enfin nous présentons une dernière variante des grammaires AB, nommée les *grammaires catégorielles avec itérations*. Cette variante, introduite dans [74], offre une représentation structurelle des liens entre constituants un peu plus juste en termes de dépendances. Concrètement, ce formalisme propose simplement des opérateurs supplémentaires pour gérer les itérations, ainsi bien sûr que les règles qui permettent de les manipuler. L'ensemble des opérateurs est $\mathcal{O}_{it} = \{ s ; / ; \backslash ; /^* ; \backslash^* ; /^+ ; \backslash^+ ; /^? ; \backslash^? \}$, et l'ensemble des règles universelles $\mathcal{R}_{it}$ est défini par :

$$\mathcal{R}_{it} = \left\{ \begin{array}{lll} (R_{FA}) & A/B \quad B \rightarrow A & (\text{avec } Var(R_{FA}) = \{A, B\}) ; \\ (R_{BA}) & B \quad B\backslash A \rightarrow A & (\text{avec } Var(R_{BA}) = \{A, B\}) ; \\ (R_{/^*_{it}}) & A/^*B \quad B \rightarrow A/^*B & (\text{avec } Var(R_{/^*_{it}}) = \{A, B\}) ; \\ (R_{/^*_{\varepsilon}}) & A/^*B \rightarrow A & (\text{avec } Var(R_{/^*_{\varepsilon}}) = \{A, B\}) ; \\ (R_{\backslash^*_{it}}) & B \quad B\backslash^*A \rightarrow B\backslash^*A & (\text{avec } Var(R_{\backslash^*_{it}}) = \{A, B\}) ; \\ (R_{\backslash^*_{\varepsilon}}) & B\backslash^*A \rightarrow A & (\text{avec } Var(R_{\backslash^*_{\varepsilon}}) = \{A, B\}) ; \\ (R_{/^+_{it}}) & A/^+B \quad B \rightarrow A/^+B & (\text{avec } Var(R_{/^+_{it}}) = \{A, B\}) ; \\ (R_{/^+_{1}}) & A/^+B \quad B \rightarrow A & (\text{avec } Var(R_{/^+_{1}}) = \{A, B\}) ; \\ (R_{\backslash^+_{it}}) & B \quad B\backslash^+A \rightarrow B\backslash^+A & (\text{avec } Var(R_{\backslash^+_{it}}) = \{A, B\}) ; \\ (R_{\backslash^+_{1}}) & B \quad B\backslash^+A \rightarrow A & (\text{avec } Var(R_{\backslash^+_{1}}) = \{A, B\}) ; \\ (R_{/^?_{1}}) & A/^?B \quad B \rightarrow A & (\text{avec } Var(R_{/^?_{1}}) = \{A, B\}) ; \\ (R_{/^?_{\varepsilon}}) & A/^?B \rightarrow A & (\text{avec } Var(R_{/^?_{\varepsilon}}) = \{A, B\}) ; \\ (R_{\backslash^?_{1}}) & B \quad B\backslash^?A \rightarrow A & (\text{avec } Var(R_{\backslash^?_{1}}) = \{A, B\}) ; \\ (R_{\backslash^?_{\varepsilon}}) & B\backslash^?A \rightarrow A & (\text{avec } Var(R_{\backslash^?_{\varepsilon}}) = \{A, B\}) ; \end{array} \right\}$$

On peut bien entendu ne considérer qu'un sous-ensemble de ces règles, selon les différents types d'itérations qu'on souhaite autoriser.

4.3.3 Grammaires catégorielles de dépendances

Les grammaires catégorielles de dépendances (GCD) proposées par DIKOVSKY dans [40], [41] et présentées dans la partie 1.3.3.3 peuvent être formalisées, sous forme simplifiée, dans le modèle des grammaires combinatoires générales. Par rapport aux différentes formes de grammaires catégorielles présentées ci-dessus, Les langages des GCD présentent la particularité de n'être pas seulement algébriques.

Soit l'ensemble des opérateurs $\mathcal{O}_{GCD} = \{\ ^1(1),\ ^*(1),\ ^?(1),\ ^+(1),\ \#(1),\ /(2), \backslash(2), /\!\!/(2),\ \backslash\!\!\backslash(2),\ \swarrow(1),\ \searrow(1),\ \nwarrow(1),\ \nearrow(1)\ \}$. L'ensemble des règles universelles des GCD est défini par[11] :

$$\mathcal{R}_{GCD} = \left\{ \begin{array}{llllll} (R_L) & C & C^1\backslash\alpha & \rightarrow & \alpha & (\text{avec } Var(R_L) = \{C, \alpha\})\ ; \\ (R_I) & C & C^*\backslash\alpha & \rightarrow C^*\backslash\alpha & & (\text{avec } Var(R_I) = \{C, \alpha\})\ ; \\ (R_R) & C & C^+\backslash\alpha & \rightarrow C^*\backslash\alpha & & (\text{avec } Var(R_R) = \{C, \alpha\})\ ; \\ (R_O) & C & C^?\backslash\alpha & \rightarrow \alpha & & (\text{avec } Var(R_O) = \{C, \alpha\})\ ; \\ (R_{\Omega_*}) & C^*\backslash\alpha & \rightarrow \alpha & & & (\text{avec } Var(R_{\Omega_*}) = \{C, \alpha\})\ ; \\ (R_{\Omega_?}) & C^?\backslash\alpha & \rightarrow \alpha & & & (\text{avec } Var(R_{\Omega_?}) = \{C, \alpha\})\ ; \\ (R_{A_1}) & C & (\swarrow C)\backslash\!\!\backslash\alpha & \rightarrow & \#(\swarrow C)\ \alpha & (\text{avec } Var(R_{A_1}) = \{C, \alpha\})\ ; \\ (R_{A_2}) & C & (\searrow C)\backslash\!\!\backslash\alpha & \rightarrow & \#(\searrow C)\ \alpha & (\text{avec } Var(R_{A_2}) = \{C, \alpha\})\ ; \\ (R_{D_1}) & \#(\swarrow C) & (\nwarrow C)\backslash\alpha & \rightarrow & \alpha & (\text{avec } Var(R_{D_1}) = \{C, \alpha\})\ ; \\ (R_{D_2}) & \#(\swarrow C) & \alpha & \rightarrow & \alpha\ \#(\swarrow C) & (\text{avec } Var(R_{D_2}) = \{C, \alpha\}) \end{array} \right\}$$

Le modèle des grammaires combinatoires générales ne permet malheureusement pas en lui-même de rendre compte de toutes les subtilités des grammaires catégorielles de dépendances. En particulier, il faudrait que la règle D_2 ne puisse être appliquée que si la règle D_1 ne peut pas l'être pour que le système grammatical soit équivalent au formalisme d'origine (voir les règles présentées dans la partie 1.3.3.3). Par ailleurs la définition des types n'est pas aussi générale dans les GCL que dans les GCG. Mais cela ne présente pas de difficulté par rapport aux langages engendrés, puisque si les contraintes d'origine de formation des types sont respectées alors le langage de la grammaire n'est pas modifié par cet aspect.

Même si on perd une grande partie des spécificités des grammaires catégorielles de liens en passant à cette représentation sous forme de GCG, ce système de $\mathcal{R}$-grammaires illustre la possibilité de gérer les mécanismes de dépendances longues et/ou distantes à la manière des GCL.

4.3.4 Grammaires catégorielles de liens

Les grammaires de liens, définies par Sleator et Temperley [98] et présentées dans la partie 1.3.3.4, ont un certain nombre de points communs avec les grammaires catégorielles. Dans les deux formalismes les grammaires sont lexicalisées et relient les mots (ou les constituants pour les grammaires catégorielles) par une relation de dépendance. Cette relation présente cependant deux différences majeures entre les deux formalismes : dans les grammaires de liens la dépendance n'est pas orientée et ne s'applique qu'à des mots, tandis que dans les grammaires catégorielles celle-ci est orientée et s'applique plus généralement aux constituants.

Nous proposons ici un formalisme intermédiaire dont le principe est celui des grammaires de liens mais plus proche du formalisme des grammaires catégorielles. Dans ce formalisme, appelé *grammaires catégorielles de liens* (GCL) et exprimé dans le cadre des grammaires combinatoires générales, les dé-

[11] Nous ne donnons ici que les règles orientée à gauche, celles orientées à droite étant totalement symétrique.

pendances sont orientées et s'appliquent aux constituants de la phrase. Nous montrons que les grammaires de liens peuvent être exprimées dans ce système de $\mathcal{R}$-grammaires, au prix de quelques restrictions relativement mineures.

4.3.4.1 Dérivation avec les grammaires catégorielles de liens

Soit $\mathcal{O}_{GCL} = \{\ \texttt{d}/2,\ \texttt{cons}/2,\ \texttt{nil}/0\}$ l'ensemble des opérateurs. L'ensemble des règles universelles $\mathcal{R}_{GCL}$ est défini par les deux règles suivantes :

$$\texttt{d(L, cons(c, R)), d(cons(c, nil), nil)} \rightarrow \texttt{d(L,R)}$$
$$\texttt{d(nil, cons(c, nil)), d(cons(c, L), R)} \rightarrow \texttt{d(L,R)}$$

Dans ces règles, c est un connecteur, L et R sont des listes de connecteurs formées à l'aide des opérateurs `cons` et `nil`. Une liste de connecteurs $[c_1, c_2, .., c_n]$ est décrite par le terme

$$\texttt{cons(}c_1\texttt{,cons(}c_2\texttt{,..cons(}c_n\texttt{,nil)..))}^{12}.$$

L'ensemble des types est l'ensemble $\{\texttt{d(L,R)}\}$, où L et R sont des listes de connecteurs[13]. Bien sûr cette définition demeure cohérente avec la définition originale des types des grammaires de liens.

Exemple 4.4. Avec le lexique défini ci-dessous, la phrase "Le chat chasse un serpent" peut être représentée par la dérivation de la figure 4.5.

le, un	▷	`d([],[D])`
chat, serpent	▷	`d([D],[S])`, `d([D,O],[])`
chasse	▷	`d([S],[O])`

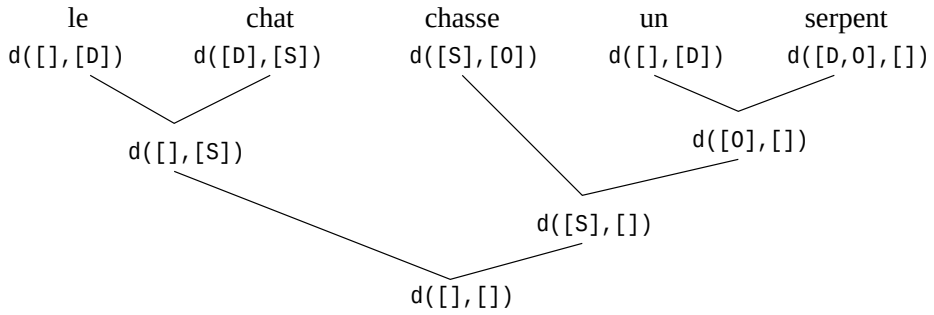


Figure 4.5 – Dérivation avec les grammaires catégorielles de liens

Remarque : un réseau de liens dans le formalisme des grammaires de liens classiques peut correspondre à plusieurs structures de dérivations équivalentes dans le cadre des GCL. Ces ambiguités apparaissent à cause de la non-orientation des liens et du fait qu'aucun ordre n'est requis entre les connecteurs.

[12] Cependant nous conserverons dans la suite la première notation, plus lisible, lorsque cela ne crée pas d'ambiguité. Il faut tout de même noter qu'il n'existe plus d'associativité dans cette forme de liste.

[13] *Remarque* : cette restriction ne contredit pas la définition générale de l'ensemble des types Tp comme l'ensemble des $\mathcal{O}_{GCL}$-termes sur Pr (avec Pr un ensemble de types primitifs, c'est-à-dire en l'occurence de connecteurs). En effet, les règles de $\mathcal{R}_{GCL}$ ne permettent pas d'utiliser des types "mal formés" de Tp.

4.3.4.2 Équivalence entre les grammaires de liens sans cycles et les GCL

La définition d'origine n'interdit pas les *cycles*, c'est-à-dire le fait qu'un mot soit connecté à deux mots (distincts) qui sont eux-mêmes reliés par un autre chemin. Par exemple, le réseau de liens de la figure 4.6 ci-dessous, dans lequel les liens (R,C,S,B) forment un cycle, est valide.

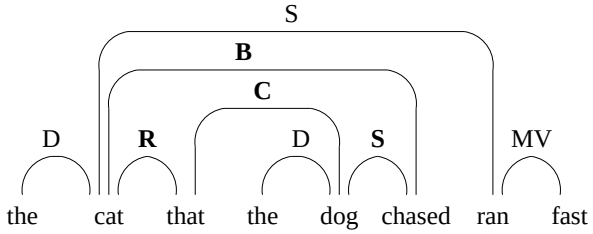


Figure 4.6 – Cycle dans les grammaires de liens

Un réseau de lien est dit *sans cycle* s'il ne contient aucun cycle. Cette définition est étendue aux grammaires de la manière suivante : une grammaire de liens est *sans cycle* si pour tout phrase correcte dans cette grammaire il existe un réseau de liens sans cycles.

Proposition 4.9. *Soit $\langle t_1, t_2, \ldots, t_n \rangle$ une séquence de types, avec $n > 1$. Les deux propositions ci-dessous sont équivalentes :*

i. Il existe un réseau de liens sans cycle valide pour la séquence $\langle t_1, t_2, \ldots, t_n \rangle$.

ii. Il existe un type t' et entier k, $1 \leq k < n$, tels que $t_k, t_{k+1} \rightarrow t'$ et il existe un réseau de liens sans cycle valide pour la séquence $\langle t_1, \ldots, t_{k-1}, t', t_{k+2}, ., t_n \rangle$.

Démonstration. (i) $\Rightarrow$ (ii). On suppose qu'il existe un réseau de liens sans cycles valide pour la séquence $\langle t_1, t_2, \ldots, t_n \rangle$. Comme le réseau de liens satisfait le critère de connectivité et ne contient aucun cycle, il existe nécessairement un type t_i qui n'est connecté que par un seul lien au reste du réseau. Soit t_j le type auquel t_i est connecté par ce lien. Si $i < j$, on pose $l = i$ et $r = j$, sinon $l = j$ et $r = i$. On montre par induction sur la distance $|r - l|$ qu'il existe deux types adjacents t_k, t_{k+1} dans l'intervalle $[t_l, t_r]$ qui sont connectés ensemble et tels que l'un d'eux n'a pas d'autre lien que celui-ci :

- Si $|r - l| = 1$ alors t_i et t_j sont adjacents. Comme t_i est connecté seulement à t_j, la propriété est vérifiée.
- Cas où $|r - l| > 1$. Tout lien depuis un type qui appartient à l'intervalle $[t_l, t_r]$ doit nécessairement être connecté à un autre type dans cet intervalle, sans quoi ce lien croiserait le lien entre t_i et t_j ce qui contredirait le critère de planarité. Comme le réseau de liens ne contient pas de cycle, il existe un type $t_{i'}$ qui n'est connecté que par un seul lien à un type $t_{j'}$, avec $l < i' < j' \leq r$ ou $l \leq j' < i' < r$. $|i' - j'| < |r - l|$, donc par hypothèse d'induction il existe deux types adjacents t_k, t_{k+1} dans l'intervalle $[t_{i'} \ldots t_{j'}]$ (ou $[t_{j'} \ldots t_{i'}]$ si $j' < i'$) qui sont connectés ensemble et tels que l'un d'eux n'est connecté que par ce lien.

Si t_k n'a qu'un seul lien qui va vers t_{k+1} (resp. t_{k+1} n'a qu'un seul lien qui va vers t_k), alors il existe un connecteur `C` tel que t_k = `d(nil, cons(C,nil))` et t_{k+1} = `d(cons(C,L),R)` (resp.

$t_k = \texttt{d(L,cons(C,R))}$ et $t_{k+1} = \texttt{d(cons(C,nil),nil)}$). Par conséquent il existe $t' = \texttt{d(L,R)}$ tel que $t_k, t_{k+1} \rightarrow t'$. t_k (resp. t_{k+1}) n'est connecté à aucun autre type que t_{k+1} (resp. t_k) et les connecteurs contenus dans les listes $\texttt{L}, \texttt{R}$ dans t_{k+1} (resp. t_k) sont les mêmes que ceux contenus dans t', donc il est possible de remplacer le couple de types (t_k, t_{k+1}) par t' dans le réseau de liens : le nouveau réseau ainsi obtenu pour la séquence $\langle t_1, \ldots, t_{k-1}, t', t_{k+2}, ., t_n \rangle$ est valide et sans cycle.

(ii) $\Leftarrow$ *(i)*. Supposons qu'il existe un réseau de liens valide et sans cycle pour la séquence de types $\langle t_1, \ldots, t_{k-1}, t', t_{k+2}, ., t_n \rangle$, et qu'il y ait deux types t_k, t_{k+1} dans cette séquence tels que $t_k, t_{k+1} \rightarrow t'$. D'après la forme des règles qui définissent la relation $\rightarrow$, il est clair que l'un des deux types ne peut avoir qu'un seul connecteur C, relié à l'autre type. Les autres connecteurs de ce couple de types sont nécessairement les mêmes que ceux contenus dans t', il est donc possible de remplacer t' dans le réseau de liens par t_k et t_{k+1}, connectés ensemble par un lien C. Ce remplacement ne crée pas de cycles, donc le réseau de liens obtenu pour $\langle t_1, \ldots, t_k, t_{k+1}, \ldots, t_n \rangle$ est valide et sans cycle.

□

Proposition 4.10. *Soit $\langle t_1, t_2, \ldots, t_n \rangle$ une séquence de types, avec $n \geq 1$. Il existe un réseau de liens valide et sans cycle pour cette séquence si et seulement si $t_1, t_2, \ldots, t_n \Rightarrow^* \texttt{d(nil,nil)}$.*

Démonstration. Par induction sur la longueur n de la séquence :

- Si $n = 1$, t_1 est nécessairement $\texttt{d(nil, nil)}$ car c'est le seul réseau de liens valide ne comportant qu'un seul type. $\Rightarrow^*$ est réflexive.
- Si $n > 1$. D'après la proposition 4.9, il existe un réseau de liens valide et sans cycle pour $\langle t_1, t_2, \ldots, t_n \rangle$ si et seulement si il existe deux types adjacents (t_i, t_{i+1}) $(0 < i < n)$ et un type t' tels que $t_i t_{i+1} \rightarrow t'$ (1), et il existe un réseau de liens valide et sans cycle pour $\langle t_1 \ldots t_{i-1} t' t_{i+2} \ldots t_n \rangle$ (2). La longueur de cette séquence de types est $n - 1$, donc par hypothèse d'induction la propriété (2) est vérifiée si et seulement si $t_1 \ldots t_{i-1} t' t_{i+2} \ldots t_n \Rightarrow^* \texttt{d(nil,nil)}$. Or $t_i t_{i+1} \rightarrow t'$ (1), donc la définition de $\Rightarrow$ donne $t_1 t_2 \ldots t_n \Rightarrow t_1 \ldots t_{i-1} t' t_{i+2} \ldots t_n \Rightarrow^* \texttt{d(nil,nil)}$, ce qui est équivalent à $t_1 t_2 \ldots t_n \Rightarrow^* \texttt{d(nil,nil)}$.

□

Corollaire 4.1. *Les grammaires de liens sans cycle et les grammaires catégorielles de liens sont faiblement équivalentes.*

Il est évident qu'un réseau de liens sans cycle forme un arbre, de même que la structure de dérivation correspondante dans le cadre des grammaires catégorielles de liens. Mais il n'y a pas d'isomorphisme entre ces arbres, du fait que les nœuds sont les mots dans la première structure tandis qu'ils correspondent à des constituants dans la seconde. Néanmoins la relation entre les deux structures est plus étroite qu'une simple équivalence faible, parce que chaque nœud dans l'arbre de dérivation correspond nécessairement à un lien dans le réseau sans cycle.

Cette "ressemblance" entre les structures est importante car elle implique que le sens de chaque connecteur peut être préservé lorsqu'une grammaire de liens est convertie en grammaire catégorielle de liens. Ainsi un arbre de dérivation de la grammaire CLG contruite décrit les mêmes relations syntaxiques que la grammaire de liens d'origine.

Cette possibilité de transformer une grammaire de liens en GCL de façon directe et relativement fiable nous sera très utile pour les applications que nous proposons dans le chapitre 7.

4.4 Apprenabilité des GCG rigides à partir de structures

KANAZAWA a proposé dans [60] une généralisation simple de l'algorithme RG à sa version des grammaires combinatoires générales, à savoir restreinte aux grammaires algébriques lexicalisées (présentée brièvement dans la partie 3.4.2). Nous étendons ici cette généralisation de façon à l'adapter à notre nouvelle définition des grammaires combinatoires générales, et démontrons de la même manière que KANAZAWA la convergence d'un tel algorithme.

4.4.1 Définitions

Nous reprenons ici quelques unes des définitions et propositions données dans la partie 3.2.3 pour le cas particulier des grammaires AB. Les notations sont conservées dans cette version qui est généralisée à l'ensemble des $\mathcal{R}$-grammaires.

Proposition 4.11. . *Soient G_1 et G_2 deux $\mathcal{R}$-grammaires dans le système $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*
Si $\sigma[G_1] \subseteq G_2$ alors $SL(G_1) \subseteq SL(G_2)$.

Démonstration. On montre d'abord que si $t_1 \dots t_n \rightarrow_{G_1} u_1 \dots u_m$ alors on a aussi $\sigma(t_1) \dots \sigma(t_n) \rightarrow_{G_2} \sigma(u_1) \dots \sigma(u_m)$ (avec $\sigma(t_i) = t_i$ si $t_i \in \Sigma$) :

- si $t_1 \dots t_n \triangleright_{G_1} u_1 \dots u_m$ (règle locale), alors par définition $\sigma(t_1) \dots \sigma(t_n) \rightarrow_{G_2} \sigma(u_1) \dots \sigma(u_m)$ est une règle locale de G_2 puisque $\sigma[G_1] \subseteq G_2$.
- si $t_1 \dots t_n \rightarrow_R u_1 \dots u_m$ (règle universelle), avec $R \in \mathcal{R}$. Soit $R = A_1 \dots A_n \rightarrow B_1 \dots B_m$, il existe une substitution τ_1 telle que $\tau_1(A_i) = t_i$ pour tout i et $\tau_1(B_j) = u_j$ pour tout j. Soit $\tau_2 = \sigma \circ \tau_1$: par définition $\tau_2(A_i) = \sigma(t_i)$ pour tout i et $\tau_2(B_j) = \sigma(u_j)$ pour tout j, donc $\sigma(t_1) \dots \sigma(t_n) \rightarrow_R \sigma(u_1) \dots \sigma(u_m)$.

Soit $S \in SL(G_1)$ une $\mathcal{R}$-structure, et P_1 une instance de S pour G_1. Soit $P_2 = \sigma[P_1]$ la structure de dérivation obtenue à partir de P_1 en remplaçant à chaque nœud le type t de ce nœud par $\sigma(t)$. $\sigma[G_1] \subseteq G_2$ et on a vu que si $t_1 \dots t_n \rightarrow_{G_1} u_1 \dots u_m$ alors $t'_1 \dots t'_n \rightarrow_{G_2} \sigma(u_1) \dots \sigma(u_m)$, avec $t'_i = t_i$ si $t_i \in \Sigma$ et $t'_i = \sigma(t_i)$ sinon. Ainsi tout nœud de P_2 est valide pour G_2, donc P_2 est une instance de T pour G_2, ce qui signifie que $T \in SL(G_2)$.

□

Deux règles $r = a_1 \dots a_n \triangleright b_1 \dots b_m$ et $r' = a'_1 \dots a'_{n'} \triangleright b'_1 \dots b'_{m'}$ sont différentes ($r \neq r'$) si au moins une des conditions suivantes est remplie :

- $n \neq n'$;
- $m \neq m'$;
- il existe i, $1 \leq i \leq n$, tel que $a_i \neq a'_i$;
- il existe i, $1 \leq i \leq m$, tel que $b_i \neq b'_i$.

Définition 4.24 (Substitution fidèle)**.** Une substitution σ est *fidèle* à une grammaire $G = \langle \Sigma, Pr, \triangleright \rangle$ si $\sigma(r) = \sigma(r')$ implique $r = r'$ pour tout couple de règles locales r, r' dans G.

Comme on peut le voir dans cette définition, une substitution n'est fidèle à une grammaire que si cette substitution ne permet pas de rendre identiques deux règles distinctes de la grammaire.

Définition 4.25 ($\sqsubseteq$)**.** Soit $\sqsubseteq$ la relation binaire sur les grammaires définie par $G \sqsubseteq G'$ si et seulement s'il existe une substitution σ telle que

- σ est fidèle à G ;
- $\sigma[G] \subseteq G'$.

La relation $\sqsubset$ est définie comme l'ordre strict correspondant à la relation $\sqsubseteq$, c'est-à-dire $G \sqsubset G'$ si et seulement si $G \sqsubseteq G'$ et $G \neq G'$. Rappelons que $G \equiv G'$ si et seulement si G et G' ne diffèrent que par un renommage de variables (définition 4.13). De telles grammaires ont des langages (de $\mathcal{R}$-structures et de chaînes) identiques.

Proposition 4.12. *La relation $\sqsubseteq$ est un ordre partiel.*

Démonstration. Montrons que $\sqsubseteq$ est un ordre partiel, c'est-à-dire est une relation réflexive, antisymétrique et transitive :

- $\sqsubseteq$ est réflexive : toute grammaire G vérifie $\sigma[G] \subseteq G$, avec σ la substitution identité. Cette substitution est trivialement fidèle.
- $\sqsubseteq$ est antisymétrique : supposons que $G_1 \sqsubseteq G_2$ et $G_2 \sqsubseteq G_1$, c'est-à-dire qu'il existe σ_1 et σ_2 fidèles telles que $\sigma_1[G_1] \subseteq G_2$ et $\sigma_2[G_2] \subseteq G_1$. Ainsi $\sigma_2[\sigma_1[G_1]] \subseteq \sigma_2[G_2] \subseteq G_1$, donc le nombre de règles dans G_1 et G_2 est identique, ce qui implique $G_1 = \sigma_2[G_2]$ et $G_2 = \sigma_1[G_1]$. Ainsi $G_1 = \sigma_2[G_2] = \sigma_2[\sigma_1[G_1]]$, donc $\sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1 = Id$, autrement dit $\sigma_1 = \sigma_2^{-1}$ et réciproquement : les deux substitutions sont l'inverse l'une de l'autre. Or cela n'est possible que dans le cas où σ_1 et σ_2 sont de simples renommages des variables. Par conséquent $G_1 = G_2$ (modulo un renommage des variables).
- $\sqsubseteq$ est transitive : si $G_1 \subseteq G_2$ et $G_2 \subseteq G_3$ alors il existe deux substitutions σ_1 et σ_2 telles que $\sigma_1[G_1] \subseteq G_2$ et $\sigma_2[G_2] \subseteq G_3$. Ceci implique que $\sigma_2[\sigma_1[G_1]] \subseteq \sigma_2[G_2] \subseteq G_3$, donc il existe une substitution $\sigma_3 = \sigma_2 \circ \sigma_1$ telle que $\sigma_3[G_1] \subseteq G_3$. σ_1 et σ_2 sont fidèles, donc pour tout couple de règles $r, r' \in G_1$ telles que $r \neq r'$ $\sigma_1(r) \neq \sigma_1(r')$ et $\sigma_2(\sigma_1(r)) \neq \sigma_2(\sigma_1(r'))$, soit $\sigma_3(r) \neq \sigma_3(r')$. Par conséquent σ_3 est fidèle.

□

Proposition 4.13. *Soient G_1 et G_2 deux $\mathcal{R}$-grammaires dans le système $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*
Si $G_1 \sqsubseteq G_2$ alors $SL(G_1) \subseteq SL(G_2)$.

Démonstration. Conséquence évidente de la proposition 4.11. □

Proposition 4.14. *Soient G_1 et G_2 deux $\mathcal{R}$-grammaires dans le système $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*
Si $G_1 \sqsubseteq G_2$ alors $|G_1| \leq |G_2|$.

Démonstration. $G_1 \sqsubseteq G_2$ donc il existe σ telle que $\sigma[G_1] \subseteq G_2$. Il est clair que $|G_1| = |\sigma[G_1]|$ par définition, or $\sigma[G_1] \subseteq G_2$ implique que $|\sigma[G_1]| \leq |G_2|$. □

Proposition 4.15. *Si $G_1 \sqsubseteq G_2$ alors $\|G_1\| \leq \|G_2\|$.*

Démonstration. $G_1 \sqsubseteq G_2$ donc il existe une substitution fidèle σ telle que $\sigma[G_1] \subseteq G_2$. Pour tout type $t \in Tp$, $\|t\| \leq \|\sigma(t)\|$, donc par définition $\|r\| \leq \|\sigma(r)\|$ pour toute règle $r \in G_1$. Comme σ est fidèle, σ définit une bijection entre les ensembles de règles de G_1 et de $\sigma[G_1]$ (toute règle $\sigma(r)$ dans $\sigma[G_1]$ a un antécédent unique r dans G_1) : ainsi il est clair que $\|G_1\| \leq \|\sigma[G_1]\|$, et comme $\sigma[G_1] \subseteq G_2$ on obtient bien $\|G_1\| \leq \|G_2\|$. □

Corollaire 4.2. *Soit $\mathcal{G}_\mathcal{R}$ une classe de $\mathcal{R}$-grammaires et $G \in \mathcal{G}_\mathcal{R}$ une $\mathcal{R}$-grammaire.*
L'ensemble $\{ G' \in \mathcal{G}_\mathcal{R} \mid G' \sqsubseteq G \}$ est fini.

Démonstration. D'après la proposition 4.15, $\{ G' \mid G' \sqsubseteq G \} \subseteq \{ G' \mid \|G'\| \leq \|G\| \}$. Or l'ensemble $\{ G' \mid \|G'\| \leq \|G\| \}$ est fini, parce que pour tout $n \in \mathbb{N}$ il n'y a qu'un nombre fini de grammaires non équivalentes entre elles dont la taille est exactement n. Par conséquent l'ensemble $\{ G' \mid G' \sqsubseteq G \}$ est également fini. □

4.4.2 Algorithme SG

Nous présentons ci-dessous une version généralisée de l'algorithme **RG** (pour *Rigid Grammars*) proposé par KANAZAWA dans [60], qui était lui-même une généralisation de l'algorithme proposé par BUSZKOWSKI et PENN dans [27]. Comme on l'a vu dans la partie 3.4.2, KANAZAWA a adapté l'algorithme de BUSZKOWSKI pour les grammaires AB rigides aux grammaires combinatoires générales rigides, mais seulement dans le cas des grammaires algébriques et lexicalisées. Nous franchissons ici encore un nouveau pas en termes de généralisation de l'algorithme de départ. Cette généralisation comporte deux aspects :

- Du point de vue algorithmique, nous prenons en compte des $\mathcal{R}$-grammaires non algébriques et/ou non lexicalisées. Une fois posée la définition généralisée de $\mathcal{R}$-structure (voir la partie 4.2), une telle adaptation est relativement simple : il s'agit simplement de quelques changements techniques, principalement au niveau de la phase d'étiquetage des structures.
- Du point de vue conceptuel, cela implique de redéfinir la notion de rigidité, de façon à gérer ces nouvelles formes de grammaires. En effet, la rigidité classique ne s'applique qu'à des grammaires lexicalisées algébriques, dans lesquelles toutes les règles sont de la forme $w \triangleright t$ (avec $w \in \Sigma$ et $t \in Tp$). Nous verrons que la définition étendue que nous proposons préserve les propriétés indispensables au bon fonctionnement de l'algorithme (et donc à l'apprenabilité des classes de langages concernées). Le principe général de l'unification est conservé : deux règles locales sont unifiées[14] si et seulement si ce couple de règles contredit à la rigidité de la grammaire. Notons qu'il

[14] On trouve aussi dans la littérature le terme *fusionner* comme synonyme du mot *unifier* dans ce contexte, notamment dans le domaine de l'inférence de langages réguliers où le premier est plus souvent employé (fusion de deux états d'un automate).

serait envisageable d'utiliser des critères de distinction des règles différents, comme le montrent par exemple BESOMBES et MARION dans [18] avec un critère de réversibilité[15].

Comme dans l'algorithme RG et la généralisation de KANAZAWA, l'algorithme d'apprentissage que nous proposons ici prend en entrée un ensemble fini de $\mathcal{R}$-structures D et renvoie une grammaire rigide (voir partie 4.4.2.2).

4.4.2.1 Grammaire générale

La première phase de l'algorithme consiste à calculer la *grammaire générale*, notée $GF(D)$ (pour *General Form grammar*). Il s'agit d'une grammaire rassemblant exactement toutes les "informations" contenues dans les structures, sans y appliquer de traitement particulier. Cette tâche est réalisée en deux phases : les structures données dans l'échantillon D en entrée sont d'abord *étiquetées*, c'est-à-dire qu'on calcule pour chacune une structure de dérivation générique (elle contient des variables), dans laquelle la forme des types est déduite des règles universelles utilisées. Cette partie est réalisée par la procédure $IG(S)$ (pour "Instance Générique") ci-dessous. Ensuite la grammaire générale est construite en créant une nouvelle règle pour chaque nœud local de chaque structure étiquetée.

Le principe de l'algorithme d'étiquetage est d'associer à chaque arc de la $\mathcal{R}$-structure S un type de manière à former une structure de dérivation P qui soit une instance de S. Les types sont formés à partir d'un ensemble de variables "fraîches", en tenant compte des contraintes imposées par les règles universelles utilisées dans la structure[16].

Dans ce but, la sortie de la structure S est d'abord étiquetée par le type s, puisque par définition si S appartient au langage de structures d'une grammaire alors sa sortie est unique et correspond au type s (associé aux phrases correctes) dans toute instance de S. On effectue aussi un premier étiquetage f_g (g pour *générique*) de la structure en associant à chaque arc e_i de celle-ci une nouvelle variable v_i. Il est important de noter que f_g est une fonction bijective, et n'est pas en général un étiquetage correct de S. Ensuite pour chaque nœud universel on récupère ses séquences d'entrées et de sorties (les variables précédemment associées aux arcs entrants/sortants du nœud). On crée aussi les séquences de types $\langle \sigma_N(A_1) \dots \sigma_N(A_m) \rangle$ et $\langle \sigma_N(B_1) \dots \sigma_N(B_{m'}) \rangle$ qui sont des copies[17] des séquences de types de la règle elle-même (la "copie" est réalisée dans l'algorithme via le renommage[18] σ_N). Pour chaque entrée (resp. sortie) du nœud un couple de types A est créé, constitué de la variable associée à cet arc ainsi que du type provenant de la règle $\sigma_N(A_i)$ (resp. $\sigma_N(B_j)$). L'union de tous ces couples de types pour toutes les entrées/sorties de tous les nœuds forme la famille d'ensemble de types $\mathcal{A}$. L'unifieur le plus général de

[15] Ce type de critère nécessite cependant une gestion algorithmique plus fine de l'unification que la rigidité, car l'unification de deux règles n'est pas toujours immédiate [17], [19]. Ceci introduit plus de souplesse et autorise la prise en compte de classes de langages différentes. La question de savoir s'il est possible de gérer les deux types de critères (rigidité et réversibilité) dans un même algorithme est laissée ouverte.

[16] Rappelons que chaque nœud correspondant à une règle universelle dans une $\mathcal{R}$-structure est étiqueté par un identifiant de cette règle.

[17] La copie est nécessaire afin que l'utilisation d'une même règle universelle en deux nœuds distincts ne provoque pas une unification erronnée des variables.

[18] σ_N est une substitution bijective, donc un renommage (voir définition 1.7).

IG(S)

Soit $E = \{e_1, \ldots, e_n\}$ l'ensemble des arcs de la $\mathcal{R}$-structure S
Soit $\mathcal{V} = \{v_1, \ldots, v_n\}$ un ensemble de n nouvelles variables
Soit f_g la fonction d'étiquetage définie par $f_g(e_i) = v_j$ si et seulement si $i = j$
Soit e_r l'unique arc en sortie de la $\mathcal{R}$-structure S
Soit $v_r \in \mathcal{V}$ l'unique variable telle que $f_g(e_r) = v_r$
$\mathcal{A} \leftarrow \{\{v_r, s\}\}$
Pour chaque nœud N représentant une règle universelle dans S **faire**
  Soit $\langle a_1, \ldots, a_m \rangle$ la séquence des arcs entrants du nœud N
  Soit $\langle b_1, \ldots, b_{m'} \rangle$ la séquence des arcs sortants du nœud N
  Pour tout $1 \leq i \leq m$, soit $t_i \in \mathcal{V}$ l'unique variable telle que $f_g(a_i) = t_i$
  Pour tout $1 \leq j \leq m'$, soit $u_j \in \mathcal{V}$ l'unique variable telle que $f_g(b_j) = u_j$
  Soit $R \in \mathcal{R}$ la règle universelle étiquetant le nœud N
  Soit $R = A_1 \ldots A_m \rightarrow B_1 \ldots B_{m'}$ et $Var(R) = \{v_1^R, \ldots, v_p^R\}$
  Soit $\mathcal{V}_N = \{v_1^N, \ldots, v_p^N\}$ un ensemble de p nouvelles variables
  Soit $\sigma_N : Var(R) \mapsto \mathcal{V}_N$ la substitution définie par $v_i^R \mapsto v_j^N$ ssi $i = j$
  Pour i **de** 1 **à** m **faire**
    $A \leftarrow \{t_i, \sigma_N(A_i)\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{A\}$
  Fin Pour
  Pour j **de** 1 **à** m' **faire**
    $A \leftarrow \{u_j, \sigma_N(B_j)\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{A\}$
  Fin Pour
Fin Pour
$\sigma_{mgu} \leftarrow$ **MGU(**$\mathcal{A}$**)**
Renvoyer $\sigma_{mgu}[f_g[S]]$

ALG. 4.1 – Étiquetage d'une $\mathcal{R}$-structure

$\mathcal{A}$ est alors calculé, ce qui permet de propager toutes les contraintes sur la forme des types (imposées par les règles universelles) à l'ensemble des variables associées aux arcs (notons que les variables internes à la structures peuvent être présentes deux fois dans $\mathcal{A}$, comme entrée et comme sortie). En appliquant cet unifieur σ_{mgu} sur la structure obtenue par l'étiquetage générique f_g sur S, on obtient une structure de dérivation minimale, complète, correcte et instance de S. Ceci sera démontré de manière plus détaillée dans la proposition 4.18.

Remarque : Le calcul d'un unifieur le plus général (MGU, défini dans la partie 1.2.2.2) ne renvoie pas toujours un résultat. Dans tous les algorithmes ci-dessous, on considère qu'en cas d'échec du calcul d'un MGU l'algorithme échoue également (sauf précision explicite contraire). De même, on considère qu'un algorithme utilisant un sous-algorithme qui renvoie un échec échoue aussi.

GF(D**)**
Soit $D = \{S_1, S_2, \ldots, S_n\}$
$G \leftarrow \emptyset$
Pour chaque $\mathcal{R}$-structure S_i dans D **faire**
  Si ($IG(S_i)$ existe) **Alors**
    Soit $P_i \leftarrow IG(S_i)$
    Pour chaque nœud N représentant une règle locale dans P_i **faire**
      Soient $\langle a_1, \ldots, a_m \rangle$ la séquence des entrées du nœud N
      Soient $\langle b_1, \ldots, b_{m'} \rangle$ la séquence des sorties du nœud N
      $G \leftarrow G \cup \{a_1 \ldots a_m \triangleright b_1 \ldots b_{m'}\}$
    Fin Pour
  Sinon
    Échec
  Fin Si
Fin Pour

ALG. 4.2 – Calcul de la grammaire générale $GF(D)$

Le calcul de la grammaire générale $GF(D)$ consiste simplement à collecter toutes les règles locales de toutes les structures de dérivation obtenues par étiquetage des $\mathcal{R}$-structures données en entrée dans D : pour chaque nœud correspondant à une règle locale la règle est ajoutée à la grammaire, constituée des types et/ou mots de Σ associés aux arcs entrants et sortants du nœud.

4.4.2.2 La grammaire synthétisée $SG(D)$

Comme dans RG, la seconde phase de l'algorithme doit unifier certaines règles de façon à produire une grammaire dont la taille n'augmente pas indéfiniment. La version que nous présentons utilise la notion étendue de rigidité des grammaires, définie ci-dessous.

Définition 4.26 (Relation $\sim_r$)**.** Soient $r = a_1 \dots a_n \triangleright b_1 \dots b_m$ et $r' = a'_1 \dots a'_{n'} \triangleright b'_1 \dots b'_{m'}$ deux règles locales quelconques. La relation $\sim_r$ est définie par $r \sim_r r'$ si et seulement si les trois conditions ci-dessous sont toutes vérifiées

- $n = n'$;
- $m = m'$;
- Pour tout i, $1 \leq i \leq n$, si $a_i \in \Sigma$ ou $a'_i \in \Sigma$ alors $a_i = a'_i$.

Définition 4.27 (Grammaire rigide)**.** Une grammaire G est *rigide* si et seulement si $r \sim_r r'$ implique $r = r'$ pour tout couple de règles locales r, r' dans G.

À partir de cette définition on constate qu'une grammaire est rigide si toute règle locale est unique par la séquence et la position des mots terminaux qu'elle contient. Ceci autorise donc des règles sans aucun terminal, mais celles-ci doivent alors se distinguer les unes des autres par le couple (n, m) (respectivement le nombre de types en partie gauche et droite). On peut également avoir un même mot terminal dans différentes règles, pourvu que celles-ci se distinguent soit par le couple (n, m), soit par la position du mot en question, ou encore par la présence de mots terminaux différents dans la règle.
Remarque : On vérifie aisément que cette définition inclut le cas de la rigidité classique, à savoir celle des grammaires algébriques lexicalisées. En effet, dans ce cas toute règle est de la forme $w \triangleright t$ avec $w \in \Sigma$ et $t \in Tp$. Ainsi pour tout couple de règles $w \triangleright t = a_1 \dots a_n \triangleright b_1 \dots b_m$ et $w' \triangleright t' = a'_1 \dots a'_{n'} \triangleright b'_1 \dots b'_{m'}$ on a toujours $n = n' = 1$ et $m = m' = 1$ d'une part, et $a_1 = w \in \Sigma$ et $a'_1 = w' \in \Sigma$ d'autre part. Par conséquent $r \sim_r r'$ si et seulement si $w = w'$. On obtient donc qu'une grammaire algébrique lexicalisée est rigide si et seulement si $w = w'$ implique $t = t'$ pour tout couple de règles $w \triangleright t$ et $w' \triangleright t'$, autrement dit il n'existe qu'une seule règle pour chaque mot du vocabulaire Σ.

Proposition 4.16. *La relation $\sim_r$ a les propriétés suivantes :*

- *$\sim_r$ est une relation d'équivalence ;*
- *Pour tout couple de règles r, r' et toute substitution σ, $r \sim_r r'$ si et seulement si $\sigma(r) \sim_r \sigma(r')$.*

Démonstration. On peut facilement vérifier que $\sim_r$ est une relation réflexive, symétrique et transitive, donc une relation d'équivalence.

Soient $r = a_1 \dots a_n \triangleright b_1 \dots b_m$ et $r' = a'_1 \dots a'_{n'} \triangleright b'_1 \dots b'_{m'}$ deux règles locales quelconques et $\sigma : \mathcal{V} \mapsto Tp$ une substitution. Rappelons que $\sigma(r) = \sigma(a_1) \dots \sigma(a_n) \triangleright \sigma(b_1) \dots \sigma(b_m)$ et $\sigma(r') = \sigma(a'_1) \dots \sigma(a'_{n'}) \triangleright \sigma(b'_1) \dots \sigma(b'_{m'})$.

- Montrons que $r \sim r'$ implique $\sigma(r) \sim_r \sigma(r')$: $r \sim_r r'$ donc $n = n'$ et $m = m'$. Si $a_i \in \sigma$ ou $a'_i \in \Sigma$ alors $a_i = a'_i$ et $\sigma(a_i) = a_i = a'_i = \sigma(a'_i)$ (car tout élément de Σ est traité comme une constante par la substitutution σ). À l'inverse, on constate qu'il est impossible que $\sigma(a_i)$ appartienne à Σ si a_i n'appartient pas à Σ, puisque l'ensemble d'arrivée de la substitution est un ensemble de types Tp (et $Tp \cap \Sigma = \emptyset$). Par conséquent les trois critères de la relation $\sim_r$ sont vérifiés, d'où $\sigma(r) \sim_r \sigma(r')$.
- Montrons que $r \not\sim_r r'$ implique $\sigma(r) \not\sim_r \sigma(r')$: si $n \neq n'$ ou $m \neq m'$, il est clair que $\sigma(r) \not\sim_r \sigma(r')$. Il reste ensuite le cas où il existe un i tel que a_i (resp. a'_i) appartienne à Σ mais $a_i \neq a'_i$: dans ce cas $\sigma(a_i) \in \Sigma$ (resp. $\sigma(a'_i) \in \Sigma$). Si $a'_i \in \Sigma$ (resp. $a_i \in \Sigma$), alors $\sigma(a_i) \neq \sigma(a'_i)$ puisque

$a_i \neq a'_i$ et $\sigma(a_i) = a_i$ et $\sigma(a'_i) = a'_i$. Sinon $a'_i \notin \Sigma$ (resp. $\sigma(a_i) \notin \Sigma$), donc $\sigma(a'_i)$ (resp. $\sigma(a_i)$) n'appartient pas à Σ non plus (cf. cas précédent), or $a_i \in \Sigma$ (resp. $a'_i \in \Sigma$), donc $\sigma(a_i) \neq \sigma(a'_i)$. Par conséquent $\sigma(r) \not\sim_r \sigma(r')$.

□

Remarque : Notons que $r = r'$ implique toujours $r \sim_r r'$, car $\sim_r$ est une relation d'équivalence (donc réflexive). On montre aisément que si G' est une grammaire rigide et si $G \subseteq G'$, alors G est aussi rigide.

La relation $\sim_r$ est une relation d'équivalence sur les règles (locales). On peut comparer cette relation à une relation de similarité, qui indique que deux règles appartenant à une même classe d'équivalence selon cette relation sont "suffisamment proches" pour être unifiées. Du fait que $\sim_r$ est une relation d'équivalence, on peut partitionner l'ensemble des règles de la grammaire générale $GF(D)$ en classes d'équivalence de telle sorte que deux règles r et r' appartiennent à une classe d'équivalence E_i si et seulement si $r \sim_r r'$. Ainsi l'algorithme crée des ensembles de règles à unifier, ces ensembles étant rassemblés dans une famille dont l'unifieur le plus général est ensuite calculé. La substitution résultante σ_{mgu} est alors appliquée à la grammaire $GF(D)$: par définition de l'unification, toutes les règles qui appartenaient à une même classe d'équivalence deviennent identiques dans $\sigma_{mgu}[GF(D)]$, donc le nombre maximal de règles de la grammaire $SG(D)$ est égal au nombre de classes d'équivalence produit par la relation $\sim_r$. Celui-ci est borné par une valeur qui dépend uniquement

- du nombre maximal d'entrée/sorties des nœuds qui apparaissent dans les $\mathcal{R}$-structures données en entrée (en effet, la longueur des règles dans $GF(D)$ ne dépend que de ce paramètre),
- et de la taille du vocabulaire Σ.

Remarque : Pour chaque règle d'une partition E_i un terme est créé : ce terme est formé avec les opérateurs "spéciaux" *regle*, *in*, *out*, les éléments de la partie gauche (mots ou types) étant placés dans l'opérateur *in*, ceux de la partie droite dans l'opérateur *out* et les deux sous-types ainsi constitués dans l'opérateur *regle*. La construction de ce terme est destinée à simplifier l'unification de règles. De façon équivalente, on aurait pu définir l'unification de deux règles $a_1 \ldots a_n \triangleright b_1 \ldots b_m$ et $a'_1 \ldots a'_{n'} \triangleright b'_1 \ldots b'_{m'}$ de la manière suivante :

1. si $n \neq n'$ ou $m \neq m'$, échec.
2. sinon, unifier la famille d'ensembles $\{ \{a_1, a'_1\}, \ldots, \{a_n, a'_{n'}\}, \{b_1, b'_1\}, \ldots, \{b_m, b'_{m'}\} \}$.

Les mots appartenant au vocabulaire Σ qui apparaissent dans ces termes sont traités comme des constantes, c'est-à-dire comme des opérateurs d'arité nulle. Cela implique que l'unification de deux mots différents produit un échec.

Comme dans l'algorithme RG, l'algorithme SG unifie des règles par calcul puis application de l'unifieur le plus général sur la famille d'ensembles définie par les partitions de la relation $\sim_r$. Ceci permet par définition d'obtenir une grammaire rigide. Contrairement au cas de l'algorithme RG, dans lequel on peut voir clairement que le nombre de règles d'une grammaire rigide est borné, la limite dépend ici de la relation $\sim_r$. Comme on le verra dans la preuve de convergence ci-dessous, c'est la propriété selon laquelle $r \sim r'$ si et seulement si $\sigma(r) \sim \sigma(r')$ qui garantit l'existence d'une limite : l'algorithme est

unif_regles(G**,** $\{E_1, \ldots, E_n\}$**)**
[$\{E_1, \ldots, E_n\}$ *est un partitionnement des règles locales de* G*]*
$\mathcal{A} \leftarrow \emptyset$
Pour i **de** 1 **à** n **faire**
  Soit $E_i = \{r_1, r_2, \ldots, r_p\}$
  $A_i \leftarrow \emptyset$
  Pour j **de** 1 **à** p **faire**
    Soit $r_j = a_1 a_2 \ldots a_m \triangleright b_1 b_2 \ldots b_{m'}$
    $A_i \leftarrow A_i \cup \{regle(in(a_1, a_2, \ldots, a_m), out(b_1, b_2, \ldots, b_{m'}))\}$
  Fin Pour
  $\mathcal{A} \leftarrow \mathcal{A} \cup A_i$
Fin Pour
Si (**MGU(**$\mathcal{A}$**)** existe) **Alors**
  $\sigma_{mgu} \leftarrow$ **MGU(**$\mathcal{A}$**)**
  Renvoyer $\sigma_{mgu}[G]$
Sinon
  Échec
Fin Si

ALG. 4.3 – unif_regles : unification des règles selon un partitionnement

SG(D**)**
$G \leftarrow$ **GF(**D**)**
Soit E l'ensemble des règles (locales) de G
Soit $\{E_1, \ldots, E_n\}$ le partitionnement de E induit par les classes d'équivalence de la relation $\sim_r$
Renvoyer unif_regles(G**,**$\{E_1, \ldots, E_n\}$**)**

ALG. 4.4 – Calcul de la grammaire synthétisée $SG(D)$

ainsi "obligé" d'unifier dès qu'elles existent deux règles qui seront à unifier, et ne peut à aucun moment unifier deux règles qui ne doivent pas l'être.

La complexité algorithmique de SG est identique à celle de l'algorithme RG, c'est-à-dire polynomiale (quadratique). Le test de la relation $\sim_r$ entre deux règles est de complexité linéaire par rapport à la taille du couple de règles. Comme dans le cas de l'algorithme RG de BUSZKOWSKI [27], on peut assez facilement modifier cet algorithme de sorte qu'il soit incrémental. Il suffit pour cela de réaliser l'unification des types entre la grammaire synthétisée obtenue sur les $i-1$ premiers exemples et la grammaire générale obtenue uniquement sur le $i^{\text{ème}}$, au lieu de recalculer l'ensemble de la grammaire générale à chaque étape. Bien sûr, la version incrémentale est préférable pour une implémentation concrète de cet algorithme.

4.4.3 Preuve de convergence

Proposition 4.17. *Soit $\mathcal{R}$ un ensemble de règles, D un ensemble fini de $\mathcal{R}$-structures et $S \in D$ une $\mathcal{R}$-structure.*

Si $IG(S)$ existe et $P = IG(S)$ alors P est une instance de S pour la grammaire $GF(D)$.

Démonstration. Clairement l'algorithme IG définit (et applique) un étiquetage sur la $\mathcal{R}$-structure donnée en entrée (voir définition 4.7). On montre que cet étiquetage est correct pour la grammaire $GF(D)$, c'est-à-dire que cet étiquetage est correct pour tout nœud N dans P :

- Si N est un nœud non étiqueté, il correspond à une règle locale. Soient $\langle a_1, \ldots, a_m\rangle$ les entrées et $\langle b_1, \ldots, b_{m'}\rangle$ les sorties de N dans P : par définition, la grammaire $GF(D)$ contient la règle $a_1 \ldots a_m \triangleright b_1 \ldots b_{m'}$, donc l'étiquetage est correct pour N.
- Soient $\langle a_1, \ldots, a_n\rangle$ (resp. $\langle b_1, \ldots, b_m\rangle$) la séquence des arcs entrants (resp. sortants) du nœud N, et $\langle t_1, \ldots, t_m\rangle$ (resp. $\langle u_1, \ldots, u_{m'}\rangle$) les variables associées à ces arcs par l'étiquetage f_g défini dans l'algorithme. Selon cet algorithme les types affectés aux arcs $a_1, \ldots, a_m, b_1, \ldots, b_{m'}$ sont respectivement $\sigma_{mgu}(t_1), \ldots, \sigma_{mgu}(t_m), \sigma_{mgu}(u_1), \ldots, \sigma_{mgu}(t_{m'})$.

  On montre que $\sigma_{mgu}(t_1) \ldots \sigma_{mgu}(t_m) \rightarrow_R \sigma_{mgu}(u_1) \ldots \sigma_{mgu}(t_{m'})$, avec R la règle utilisée au nœud N. Soit $R = A_1 \ldots A_m \rightarrow B_1 \ldots B_{m'}$ avec $Var(R) = \{v_1^R, \ldots, v_p^R\}$, et soit $\sigma_N : Var(R) \mapsto \mathcal{V}_N$ la substitution définie par $v_i^R \mapsto v_j^N$ si et seulement si $i = j$, avec $\mathcal{V}_N = \{v_1^N, \ldots, v_p^N\}$ un ensemble de p nouvelles variables. Pour tout i, $1 \leq i \leq m$ (resp. pour tout j, $1 \leq j \leq m'$), $\sigma_{mgu}(t_i) = \sigma_{mgu}(\sigma_N(A_i))$ (resp. $\sigma_{mgu}(u_j) = \sigma_{mgu}(\sigma_N(B_j))$), car σ_{mgu} est un unifieur pour chaque famille $\{t_i, \sigma_N(A_i)\}$ (resp. $\{u_j, \sigma_N(B_j)\}$). Ainsi il existe une substitution $\tau = \sigma_{mgu} \circ \sigma_N$ telle que $\tau(A_i) = \sigma_{mgu}(t_i)$ pour tout i (resp. $\tau(A_j) = \sigma_{mgu}(u_j)$ pour tout j), donc $\sigma_{mgu}(t_1) \ldots \sigma_{mgu}(t_m) \rightarrow_R \sigma_{mgu}(u_1) \ldots \sigma_{mgu}(t_{m'})$. Par conséquent l'étiquetage est correct pour N.

Ainsi P est bien une structure de dérivation pour $GF(D)$. Comme P est construite par cet étiquetage appliqué sur S, il est clair que P est une instance de S pour $GF(D)$. □

Corollaire 4.3. *Soit D un ensemble fini de $\mathcal{R}$-structures.*

Si $GF(D)$ existe alors $D \subseteq SL(GF(D))$.

Démonstration. Soit $S \in D$ une $\mathcal{R}$-structure : $GF(D)$ existe donc $IG(S)$ existe. Soit $P = IG(S)$. La proposition 4.17 indique que S est une instance de P pour $GF(D)$, autrement dit il existe une structure de dérivation P pour $GF(D)$ telle que S est une instance de P. par conséquent $S \in SL(GF(D))$. □

Proposition 4.18. *Soit G une $\mathcal{R}$-grammaire, P une structure de dérivation pour G et S une $\mathcal{R}$-structure. Si P est une instance de S alors $IG(S)$ existe et il existe une substitution σ telle que $\sigma[IG(S)] = P$.*

Démonstration. Supposons que P soit une instance de S. Par définition il existe un étiquetage f tel que $f[S] = P$. Nous reprenons les notations de l'algorithme IG :

- $\{e_1, \ldots, e_n\}$ est l'ensemble des arcs de S et $\mathcal{V} = \{v_1, \ldots, v_n\}$ l'ensemble des variables associées à ces arcs par l'étiquetage f_g.
- Pour chaque nœud N représentant une règle universelle $\langle a_1, \ldots, a_n\rangle$ (resp. $\langle b_1, \ldots, b_m\rangle$) est la séquence des arcs entrants (resp. sortants) du nœud N, et $\langle t_1, \ldots, t_m\rangle$ (resp. $\langle u_1, \ldots, u_{m'}\rangle$) est l'ensemble des variables associées à ces arcs par l'étiquetage f_g.
- Pour chaque nœud N représentant une règle universelle $R = A_1 \ldots A_m \to B_1 \ldots B_{m'}$ est cette règle, et $\mathcal{V}_N$ est une "copie" de l'ensemble des variable $Var(R)$, avec $\sigma_N : Var(R) \mapsto \mathcal{V}_N$ le renommage qui définit cette bijection.

Par définition f est un étiquetage correct pour tout nœud N de P, donc il existe une substitution σ_R telle que $\sigma_R(A_i) = f(a_i)$ pour tout i et $\sigma_R(B_j) = f(b_j)$ pour tout j. On peut ainsi définir une substitution $\tau : \mathcal{V} \cup \bigcup_{N \in S} \mathcal{V}_N \mapsto Tp_G$ par

- pour tout $v \in \mathcal{V}$, $\tau(v) = f(e)$ si et seulement si $f_g(e) = v$ (autrement dit τ associe à chaque variable v le type associé par l'étiquetage f à l'arc correspondant à v selon l'étiquetage générique f_g).
- pour tout $v \in \mathcal{V}_N$ et pour tout N dans S, $\tau(v) = \sigma_R(\sigma_N^{-1}(v))$.[19]

On montre que τ est un unifieur de la famille d'ensemble $\mathcal{A}$ définie dans l'algorithme : pour tout couple $\{t_i, \sigma_N(A_i)\}$ (resp. $\{u_j, \sigma_N(B_j)\}$) dans $\mathcal{A}$, $\tau(t_i) = f(a_i)$ (resp. $\tau(u_j) = f(b_j)$) avec a_i (resp. b_j) l'arc auquel la variable t_i (resp. u_j) est associée par l'étiquetage f_g (qui est une bijection). $\sigma_N(A_i)$ ne contient que des variables de $\mathcal{V}_N$, donc par définition $\tau(\sigma_N(A_i)) = \sigma_R(\sigma_N^{-1}(\sigma_N(A_i))) = \sigma_R(A_i) = f(a_i)$. Ainsi on obtient $\tau(t_i) = f(a_i) = \tau(\sigma_N(A_i))$, autrement dit τ est un unifieur pour le couple $\{t_i, \sigma_N(A_i)\}$.

τ est un unifieur de $\mathcal{A}$, donc l'unifieur le plus général σ_{mgu} de $\mathcal{A}$ existe, par conséquent $IG(S)$ existe. σ_{mgu} est défini comme l'unifieur le plus général de $\mathcal{A}$, donc par définition du MGU il existe une substitution σ telle que $\tau = \sigma \circ \sigma_{mgu}$. Par définition, $\tau[f_g[S]] = P$ et $\sigma_{mgu}[f_g[S]] = IG(S)$, par conséquent $\sigma[f_g[S]] = \sigma[\sigma_{mgu}[f_g[S]]] = (\sigma \circ \sigma_{mgu})[f_g[S]] = \tau[f_g[S]] = P$. □

Proposition 4.19. *Soit G une $\mathcal{R}$-grammaire et D un ensemble fini de $\mathcal{R}$-structures.*

$D \subseteq SL(G)$ si et seulement si $GF(D)$ existe et il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$.

Démonstration. ($\Rightarrow$). Supposons que $D \subseteq SL(G)$. Soit $D = \{S_1, \ldots, S_n\}$: pour tout $S_i \in D$, il existe une structure de dérivation P_i pour G telle que P_i est une instance de S_i, car $S_i \in SL(G)$. Ainsi d'après

[19] rappelons que σ_N est une substitution bijective (renommage), donc la substitution inverse σ_N^{-1} est clairement définie.

la proposition 4.18 $IG(S_i)$ existe, et il existe une substitution σ_i telle que $\sigma_i[IG(S_i)] = P_i$. $IG(S_i)$ existe pour tout i, donc par définition $GF(D)$ existe. Notons que les ensembles de départ de σ_i et σ_j sont disjoints si $i \neq j$ (car un nouvel ensemble de variables est créé pour chaque structure S_i dans $IG(S_i)$). Ainsi il est possible de définir la substitution σ par $\sigma = \bigcup_{1 \leq i \leq n} \sigma_i$.

On montre ensuite que $\sigma[GF(D)] \subseteq G$. Soit $a_1 \dots a_m \triangleright b_1 \dots b_{m'}$ une règle locale de la grammaire $GF(D)$. Par définition de cette grammaire, il existe une $\mathcal{R}$-structure $S_i \in D$ et un nœud local dans cette structure telle que les séquences $\langle a_1, \dots, a_m \rangle$ et $\langle b_1, \dots, b_{m'} \rangle$ sont respectivement les entrées et sorties du nœud correspondant dans la structure de dérivation $IG(S_i)$. $\sigma[IG(S_i)] = \sigma_i[IG(S_i)] = P_i$, par conséquent P_i contient un nœud dont les entrées et sorties sont respectivement les séquences $\langle \sigma(a_1), \dots, \sigma(a_m) \rangle$ et $\langle \sigma(b_1), \dots, \sigma(b_{m'}) \rangle$. Cela implique qu'il existe une règle locale $\sigma(a_1) \dots \sigma(a_m) \triangleright \sigma(b_1) \dots \sigma(b_{m'})$ dans G, car P_i est une structure de dérivation correcte pour G. Ceci étant valable pour toutes les règles de $GF(D)$, on obtient bien que $\sigma[GF(D)] \subseteq G$.

($\Leftarrow$). Supposons que $GF(D)$ existe et qu'il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$. Selon la proposition 4.11, $SL(GF(D)) \subseteq SL(G)$. Or $D \subseteq SL(GF(D))$ d'après le corollaire 4.3, donc $D \subseteq SL(G)$. □

Proposition 4.20. *Soit σ une substitution et D un ensemble fini de $\mathcal{R}$-structures tel que $GF(D)$ existe.*

σ est un unifieur pour la famille d'ensemble $\mathcal{A}$ (définie dans l'algorithme SG) si et seulement si $\sigma[GF(D)]$ est rigide.

Démonstration. $\Rightarrow$. Supposons que σ soit un unifieur pour $\mathcal{A} = \{A_1, \dots, A_n\}$. On montre la contraposée : supposons que $\sigma[GF(D)]$ ne soit pas rigide, c'est-à-dire qu'il existe des règles dans $GF(D)$ telles que $\sigma(r) \sim_r \sigma(r')$ et $\sigma(r) \neq \sigma(r')$. Selon la proposition 4.16, $\sigma(r) \sim_r \sigma(r')$ implique que $r \sim_r r'$. La famille $\mathcal{A}$ est définie par les classes d'équivalence de la relation $\sim_r$, donc par construction $r_1 \in A_i$ et $r_2 \in A_i$ si seulement si $r_1 \sim r_2$. Comme σ est un unifieur de $\mathcal{A}$, pour tout couple de règles $r_1, r_2 \in GF(D)$ $r_1 \sim_r r_2$ implique $\sigma(r_1) = \sigma(r_2)$ et donc en particulier $\sigma(r) = \sigma(r')$, ce qui contredit l'hypothèse selon laquelle il existe deux règles r et r' telles que $\sigma(r) \sim_r \sigma(r')$ et $\sigma(r) \neq \sigma(r')$. Par conséquent $\sigma[GF(D)]$ est rigide.

$\Leftarrow$. Supposons que $\sigma[GF(D)]$ soit une grammaire rigide. Soient r et r' deux règles de $GF(D)$ appartenant à une même famille A_i dans $\mathcal{A}$. On a montré ci-dessus que la famille $\mathcal{A}$ est construite selon les classes d'équivalences de $\sim_r$, donc les règles r, r' appartiennent à la même famille A_i si et seulement si $r \sim_r r'$. $r \sim r'$ implique $\sigma(r) \sim \sigma(r')$, d'après la proposition 4.16. Or $\sigma[GF(D)]$ est rigide, donc $\sigma(r) = \sigma(r')$. Par conséquent σ est bien un unifieur pour la famille d'ensembles $\mathcal{A}$. □

Corollaire 4.4. *Soit D un ensemble fini de $\mathcal{R}$-structures.*

Si $SG(D)$ existe, alors $SG(D)$ est une grammaire rigide.

Démonstration. Si $SG(D)$ existe, $SG(D) = \sigma[GF(D)]$ avec σ l'unifieur le plus général de $\mathcal{A}$ (par définition). Il découle directement de la proposition 4.20 que cette grammaire est rigide. □

Proposition 4.21. *Soit G une $\mathcal{R}$-grammaire rigide et D un ensemble fini de $\mathcal{R}$-structures.*
$D \subseteq SL(G)$ si et seulement si $SG(D)$ existe et $SG(D) \sqsubseteq G$.

Démonstration. $\Rightarrow$. Supposons que $D \subseteq SL(G)$. D'après la proposition 4.19, $GF(D)$ existe et il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$. La grammaire $\sigma[GF(D)]$ est rigide car G est rigide, donc selon la proposition 4.20 σ est un unifieur pour la famille d'ensemble $\mathcal{A}$ définie dans l'algorithme SG. Par conséquent l'unifieur le plus général de $\mathcal{A}$ σ_{mgu} existe (proposition 1.1), et (par définition) $SG(D)$ existe et est défini par $SG(D) = \sigma_{mgu}[GF(D)]$. σ étant un unifieur de $\mathcal{A}$, il existe une substitution τ telle que $\sigma = \tau \circ \sigma_{mgu}$. Ainsi $\tau[SG(D)] = \tau[\sigma_{mgu}[GF(D)]] = \sigma[GF(D)]$. Or $\sigma[GF(D)] \subseteq G$, donc il existe une substitution τ telle que $\tau[SG(D)] \subseteq G$. On montre que τ est une substitution fidèle : soient $r \neq r'$ deux règles distinctes de $SG(D)$. $SG(D)$ est rigide (d'après le corollaire 4.4), donc $r \not\sim_r r'$. La proposition 4.16 donne donc que $\tau(r) \not\sim_r \tau(r')$, et donc $\tau(r) \neq \tau(r')$ car $\sim_r$ est réflexive. Ainsi τ est une substitution fidèle telle que $\tau[SG(D)] \subseteq G$, autrement dit $SG(D) \sqsubseteq G$.

$\Leftarrow$. Supposons que $SG(D, \sim)$ existe et que $SG(D, \sim) \sqsubseteq G$: il existe une substitution (fidèle) τ telle que $\tau[SG(D, \sim)] \subseteq G$. $SG(D, \sim) = \sigma_{mgu}[GF(D)]$, avec σ_{mgu} l'unifieur le plus général de $\mathcal{A}$ (σ_{mgu} ainsi que $GF(D)$ existent puisque $SG(D, \sim)$ existe). Ainsi il existe une substitution $\sigma = \tau \circ \sigma_{mgu}$ telle que $\sigma[GF(D)] \subseteq G$, ce qui implique selon la proposition 4.19 que $D \subseteq SL(G)$. □

Proposition 4.22. *Soit G une $\mathcal{R}$-grammaire rigide et D_1, D_2 deux ensembles finis de $\mathcal{R}$-structures.*
Si $D_1 \subseteq D_2 \subseteq SL(G)$ alors $SG(D_1)$ et $SG(D_2)$ existent, et $SG(D_1) \sqsubseteq SG(D_2) \sqsubseteq G$.

Démonstration. Il découle de la proposition 4.21 que $SG(D_1)$ et $SG(D_2)$ existent, et que $SG(D_1) \sqsubseteq G$ et $SG(D_2) \sqsubseteq G$. $SG(D_2) \sqsubseteq SG(D_2)$ car $\sqsubseteq$ est réflexive, donc selon la proposition 4.21 (utilisée ici dans l'autre sens) on a $D_2 \subseteq SL(SG(D_2))$. Comme $D_1 \subseteq D_2 \subseteq SL(SG(D_2))$, on obtient que $SG(D_1) \sqsubseteq SG(D_2)$ en appliquant encore une fois la proposition 4.21. □

Théorème 4.1. *Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires.*
Dans le système de grammaires $\langle SL(\mathcal{R}), \mathcal{G}_\mathcal{R}, SL \rangle$, avec $\mathcal{G}_\mathcal{R}$ l'ensemble des $\mathcal{R}$-grammaires, l'algorithme $SG(D)$ apprend la classe $\{ SL(G) \mid G \in \mathcal{G}_\mathcal{R}$ est rigide $\}$ des langages de structures rigides :

Pour toute grammaire rigide $G \in \mathcal{G}_\mathcal{R}$ et toute énumération $\langle S_1, S_2, \ldots \rangle$ de $SL(G)$, il existe un entier $n \in \mathbb{N}$ tel que $SL(SG(\{S_1, S_2, \ldots, S_i\})) = SL(G)$ pour tout entier $i \geq n$.

Démonstration. Soit $D_i = \{S_1, S_2, \ldots, S_i\}$ pour tout $i \in \mathbb{N}$. Par hypothèse, $D_i \subseteq D_{i+1} \subseteq SL(G)$ pour tout $i \in \mathbb{N}$. D'après la proposition 4.22, ceci implique que $SG(D_i)$ et $SG(D_{i+1})$ existent, et que $SG(D_i) \sqsubseteq SG(D_{i+1}) \sqsubseteq G$. Or il n'y a qu'un nombre fini de grammaires G' différentes telles que $G' \sqsubseteq G$ (par le corollaire 4.2), c'est pourquoi SG converge nécessairement sur la séquence $\langle S_i \rangle_{i \in \mathbb{N}}$ vers une grammaire G_n : il existe $n \in \mathbb{N}$ tel que $SG(D_i) = G_n$ pour tout $i \geq n$. Ainsi $SL(G) = \{ S_i \mid i \in \mathbb{N} \} \subseteq SL(G_n)$, car $D_i \subseteq SL(SG(D_i)) = SL(G_n)$ pour tout $i \geq n$ (par la proposition 4.21, utilisée avec $SG(D_i) \sqsubseteq SG(D_i)$). Comme $G_n = SG(D_n) \sqsubseteq G$, on a aussi $SL(G_n) \subseteq SL(G)$ d'après la proposition 4.11. Par conséquent on a $SL(G_n) \subseteq SL(G)$ et $SL(G) \subseteq SL(G_n)$, d'où $SL(G_n) = SL(G)$. □

Ce résultat est applicable à toutes les familles de $\mathcal{R}$-grammaires, notamment celle des grammaires AB et plus généralement toutes celles des *general combinatory grammars* pour lesquelles KANAZAWA l'a déjà démontré. Notons qu'il s'applique aussi aux grammaires non algébriques : par exemple, la classe des grammaires contextuelles rigides est apprenable à partir de $\mathcal{R}$-structures.

4.4.4 Extension éventuelle aux grammaires k-bornées

De même que KANAZAWA a décrit l'algorithme VG_k servant à l'apprentissage de grammaires AB k-valuées (à partir de FA-structures), nous proposons ci-dessous l'algorithme généralisé équivalent k-SG (à la différence près que VG_k renvoie des grammaires k-valuées tandis que k-SG renvoie des grammaires k-bornées). Mais cet algorithme n'est pas un algorithme d'apprentissage (au sens de GOLD), car (comme l'algorithme VG_k) celui-ci renvoie un ensemble de grammaires et non une seule solution : il est donc en général impossible que cet algorithme converge vers *la* grammaire cible. En revanche on peut démontrer certaines propriétés de l'ensemble de grammaires fourni par k-SG, notamment le fait qu'il existe toujours dans cet ensemble une grammaire qui converge vers la grammaire cible. Cela nous sera utile en particulier dans le chapitre 6.

4.4.4.1 Algorithme

k-SG(D**)**
  $G \leftarrow$ **GF(**D**)**
  Soit E l'ensemble des règles (locales) de G
  Soit $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ l'ensemble des k-partitionnements de E
  $Res \leftarrow \emptyset$
  Pour i **de** 1 **à** n **faire**
    Si (**unif_regles(**G**,** $\mathcal{A}_i$**)** existe) **Alors**
      $Res \leftarrow Res \cup \{$**unif_regles(**$G$**,** $\mathcal{A}_i$**)** $\}$
    Fin Si
  Fin Pour
  Renvoyer Res

ALG. 4.5 – Calcul des grammaires synthétisées k-bornées k-SG(D)

Dans cet algorithme, l'ensemble de tous les k-partitionnements possibles de l'ensemble de règles de la grammaire G sont générés. On essaie ensuite de réaliser l'unification de chaque famille d'ensembles de règles ainsi constituée : bien entendu, parmi ces familles il est possible qu'un grand nombre ne soient pas unifiables[20]. Seules les familles pour lesquelles un unifieur existe sont prises en compte dans l'ensemble de grammaires solutions.

[20] En particulier, on peut obtenir des partitions contenant deux règles r, r' telles que $r \not\sim_r r'$: l'unification d'un tel ensemble de types produira nécessairement un échec. Il serait également possible de définir l'algorithme en essayant d'unifier seulement les ensembles de règles comparables par la relation $\sim_r$. Cette version se rapprocherait plus de l'algorithme VG_k proposé par KANAZAWA (en adaptant la notion de grammaire k-valuée à la relation $\sim_r$, comme pour la rigidité).

Il est important de noter que l'objectif de cet algorithme est seulement de *contenir* l'ensemble des solutions possibles. On montrera ci-dessous que, pour tout sous-ensemble D du langage d'une grammaire k-bornée G, il existe une grammaire G' dans l'ensemble de solutions de k-SG(D) telle que $G' \sqsubseteq G$.

Cet algorithme est de toute évidence de complexité exponentielle, puisqu'il est basé sur le calcul des k-partionnements de l'ensemble de règles. COSTA FLORÊNCIO montre d'ailleurs dans [33] que l'apprentissage de grammaires AB k-valuées est NP-difficile.

L'existence d'un algorithme d'apprentissage pour les grammaires k-bornées dépend en fait de la décidabilité du problème qui consiste à déterminer une grammaire minimale (en termes de langage de structures) parmi un ensemble fini de $\mathcal{R}$-grammaires k-bornés (éventuellement dans une sous-classe fixée). En effet si ce problème est décidable par une fonction f alors il suffit de définir l'algorithme d'apprentissage par f(k-SG(D)). Nous n'entrons pas ici dans les détails de ce cas car la plupart des classes de $\mathcal{R}$-grammaires ne vérifient pas ce critère. Néanmoins le lecteur intéressé trouvera dans [60] tous les détails pour le cas des grammaires AB k-valuées, ainsi que la preuve de convergence de KANAZAWA pour ce cas.

4.4.4.2 Propriétés remarquables

Indépendamment du problème de la grammaire minimale, l'algorithme k-SG a pour intérêt de restreindre au minimum l'ensemble des solutions possibles pour un algorithme d'apprentissage de grammaires k-bornées. Nous montrons ci-dessous la principale propriété de cet algorithme, qui nous sera utile dans les prochains chapitres.

Proposition 4.23. *Soit G une $\mathcal{R}$-grammaire k-bornée et D un ensemble fini de $\mathcal{R}$-structures.*
$D \subseteq SL(G)$ si et seulement s'il existe une $\mathcal{R}$-grammaire $G' \in$ k-SG(D) telle que $G' \sqsubseteq G$.

Démonstration. $\Rightarrow$. Supposons que $D \subseteq SL(G)$. Selon la proposition 4.19, il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$. Soit $\mathcal{A} = \{A_1, \ldots, A_n\}$ le partitionnement de l'ensemble des règles de $GF(D)$ tel que pour tout couple de règles r, r' de $GF(D)$: $r \in A_i$ et $r' \in A_i$ si et seulement si $\sigma(r) = \sigma(r')$. Par définition σ est un unifieur de $\mathcal{A}$, donc l'unifieur le plus général de $\mathcal{A}$ existe aussi : soit $\sigma_{mgu} = MGU(\mathcal{A})$. σ étant un unifieur de $\mathcal{A}$, par défintion il existe une substitution τ telle que $\sigma = \tau \circ \sigma_{mgu}$.

G est k-bornée, donc $\sigma[GF(D)]$ l'est aussi puisque $\sigma[GF(D)] \subseteq G$, et par conséquent le partitionnement $\mathcal{A}$ est un k-partitionnement. Donc $\mathcal{A}$ est l'un des k-partionnements utilisés dans l'algorithme k-SG(D), d'où l'existence de la grammaire $G' = \sigma_{mgu}[GF(D)] \in$ k-SG(D). Ainsi $\sigma[GF(D)] = (\tau \circ \sigma_{mgu})[GF(D)] = \tau[\sigma_{mgu}[GF(D)]] = \tau[G']$, donc $\tau[G'] \subseteq G$.

Montrons par l'absurde que τ est une substitution fidèle : on suppose qu'il existe deux règles $r \neq r' \in G'$ telles que $\tau(r) = \tau(r')$. $G' = \sigma_{mgu}[GF(D)]$, donc il existe deux règles $r_0, r'_0 \in GF(D)$ telles que $r = \sigma_{mgu}(r_0)$ et $r' = \sigma_{mgu}(r'_0)$. Or σ_{mgu} est un unifieur de $\mathcal{A}$, donc $\sigma_{mgu}(r_0) \neq \sigma_{mgu}(r'_0)$ implique que r_0 et r'_0 n'appartiennent pas à une même partition de $\mathcal{A}$. Par définition de $\mathcal{A}$, on peut en déduire que $\sigma(r_0) \neq \sigma(r'_0)$, autrement dit $(\tau \circ \sigma_{mgu})(r_0) \neq (\tau \circ \sigma_{mgu})(r'_0)$. Or par hypothèse $\tau(\sigma_{mgu}(r_0)) = \tau(r) = \tau(r') = \tau(\sigma_{mgu}(r_0))$, il y a donc contradiction.

Par conséquent τ est une substitution fidèle telle que $\tau[G'] \subseteq G$, donc $G' \sqsubseteq G$ avec $G' \in$ k-SG(D).

$\Leftarrow$. Supposons qu'il existe une $\mathcal{R}$-grammaire $G' \in$ k-SG(D) telle que $G' \sqsubseteq G$. $G' \in$ k-SG(D) suppose (par construction) qu'il existe un k-partionnement $\mathcal{A}_i$ de l'ensemble de règles de $GF(D)$ tel que σ_i soit l'unifieur le plus général de la famille d'ensembles correspondante, et $G' = \sigma_i[GF(D)]$. Or $G' \sqsubseteq G$ implique qu'il existe une substitution σ telle que $\sigma[G'] \subseteq G$, donc $\sigma[\sigma_i[GF(D)]] \subseteq G$. Ainsi il existe une substitution $\sigma \circ \sigma_i$ telle que $(\sigma \circ \sigma_i)[GF(D)] \subseteq G$, ce qui est équivalent d'après la proposition 4.19 à $D \subseteq SL(G)$[21].

□

4.5 Conclusion

L'objectif du cadre des grammaires combinatoires générales (étendu par rapport à celui proposé par KANAZAWA dans [60]), présenté dans ce chapitre, est la représentation du plus grand nombre possible de formalismes grammaticaux. Bien entendu, tous les formalismes ne se prêtent pas aux conditions requises par ce cadre : la possibilité d'exprimer les mécanismes de base du système grammatical sous forme de règles universelles à la manière des grammaires catégorielles classiques est en effet indispensable, or tous les systèmes ne le permettent pas. Nous avons vu néanmoins dans la partie 4.3 que ce modèle offre une certaine souplesse qui permet la représentation de quelques formes assez variées de systèmes grammaticaux (au prix parfois de quelques simplifications).

Malgré certaines ressemblances, il convient de distinguer les grammaires combinatoires générales des *systèmes formels élémentaires* (*Elementary Formal Systems*, *EFS*) dont les principes de bases sont définis par SMULLYAN dans [99]. L'apprenabilité des EFS de taille bornée est l'un des résultats importants obtenus par SHINOHARA à l'aide de son théorème sur la densité finie bornée (voir partie 2.5), dans [94] [10] [96], qu'on peut voir comme une généralisation du résultat d'ANGLUIN sur les langages de motifs [4]. La terminologie utilisée dans les deux systèmes peut en effet prêter à confusion : Dans [94], SHINOHARA nomme *termes* les séquences d'éléments de $\Sigma \cup X$, avec Σ un vocabulaire terminal et X un ensemble de variables (alors que dans notre terminologie il s'agit de séquences de types primitifs et/ou mots du vocabulaire). À l'inverse, une *formule atomique* ou *atome* désigne un terme de profondeur 1[22], c'est-à-dire de la forme $f(a_1, \ldots, a_n)$ où tous les a_i sont des "termes"[23]. Nous soulignons ces différences terminologiques car celles-ci pourraient masquer la différence majeure entre les grammaires combinatoires générales et les systèmes formels élémentaires : les "éléments" manipulés par ces derniers sont "plats", tandis que les types utilisés dans les GCG sont de profondeur illimitée a priori. Cette différence est extrêmement importante du point de vue de l'apprenabilité, comme on le verra dans les chapitres suivants. Notons que le résultat d'apprenabilité de SHINOHARA sur les EFS porte uniquement sur les EFS *de taille bornée* (pour plus de détails voir [94]).

Par ailleurs, il faut souligner que le résultat de la partie 4.4 démontre simplement l'apprenabilité des

[21] $GF(D)$ existe, sinon l'ensemble k-SG(D) serait vide.
[22] Dans notre terminologie.
[23] Terminologie de SHINOHARA dans [94].

langages de structures des $\mathcal{R}$-grammaires rigides (pour toute famille de $\mathcal{R}$-grammaires), et non l'élasticité finie de toute classe de langages ainsi constituée. Il existe en effet des classes de $\mathcal{R}$-grammaires rigides qui ne possèdent pas la propriété d'élasticité finie, même si elles sont apprenables à partir de $\mathcal{R}$-structures comme on vient de le voir. La classe de langages exhibée dans l'exemple 3.2, proposée par COSTA FLORÊNCIO, en est une illustration. Ceci rend difficile l'extension de l'apprenabilité aux classes de langages des grammaires k-valuées (ou k-bornées), du moins selon la même méthode que celle de KANAZAWA pour les grammaires AB (voir partie 3.3). Or le passage des grammaires rigides aux grammaires k-valuées (ou k-bornées) est quasiment indispensable lorsqu'on envisage de traiter des langues naturelles, pour les raisons évoquées dans la partie 3.2.4. C'est pourquoi nous chercherons dans les chapitres 5 et 6 des réponses à la question laissée ouverte par KANAZAWA : quelles conditions doivent respecter l'ensemble des règles universelles d'une famille de $\mathcal{R}$-grammaires pour que celle-ci ait l'élasticité finie ?

CHAPITRE 5

Apprenabilité des GCG à arguments bornés

L'APPRENABILITÉ DES GCG RIGIDES À PARTIR DE STRUCTURES PRÉSENTE LE MÊME TYPE DE CONTRAINTES QUE CELLE DES GRAMMAIRES AB : CLASSES DE LANGAGES LIMITÉES, NÉCÉSSITÉ DE STRUCTURES COMPLEXES EN ENTRÉES. C'EST LA RAISON POUR LAQUELLE IL EST PERTINENT D'ÉTUDIER LA QUESTION DE L'ÉLASTICITÉ FINIE DES GCG, CAR CELA PERMETTRA DE DÉCOUVRIR DES SOUS-CLASSES APPRENABLES PLUS RICHES GRÂCE À LA TECHNIQUE PROPOSÉE PAR KANAZAWA. NOUS ABORDONS CETTE QUESTION AVEC LES OUTILS MIS EN PLACE PAR SHINOHARA, À SAVOIR LA DENSITÉ FINIE BORNÉE. DANS UN PREMIER TEMPS NOUS PROPOSONS UNE PETITE EXTENSION DE CE CONCEPT, AFIN DE FAIRE CONVERGER LES MÉTHODES DE DÉMONSTRATION D'APPRENABILITÉ DÉVELOPPÉES PAR SHINOHARA ET KANAZAWA. ENSUITE NOUS MONTRERONS L'ÉLASTICITÉ FINIE DE LA CLASSE DES GCG "À ARGUMENTS BORNÉS". IL S'AGIT D'UN CRITÈRE QUI A POUR AVANTAGE DE CONCERNER DE NOMBREUX FORMALISMES GRAMMATICAUX, MAIS QUI A POUR INCONVÉNIENT D'IMPOSER UNE CONTRAINTE PUREMENT TECHNIQUE POUR OBTENIR L'ÉLASTICITÉ FINIE.

5.1 Introduction

Après avoir présenté quelques formalismes grammaticaux adaptés à la représentation des langues naturelles, décrit les enjeux, techniques et résultats de l'apprenabilité dans le modèle de GOLD et proposé les grammaires combinatoires générales comme cadre uniforme de représentation des systèmes grammaticaux, nous commençons dans ce chapitre la recherche de critères impliquant l'apprenabilité, pour des classes de GCG aussi riches que possible (en termes de représentation des langues naturelles).

Nous avons montré dans le chapitre précédent que toute classe de $\mathcal{R}$-grammaires $\sim_r$-rigides est apprenable à partir de $\mathcal{R}$-structures. Mais ce résultat est soumis aux mêmes limitations que celles des grammaires AB dans le cas rigide à partir de FA-structures, décrites dans la partie 3.2.4 : d'une part la nécessité de structures complexes en entrées, difficilement disponibles pour de gros volumes de données, et d'autre part la contrainte de ($\sim_r$-)rigidité, qui limite fortement la forme des grammaires (et donc des langages) concernées. Ces deux limitations sont particulièrement gênantes pour le cas de l'apprentissage (au sens du modèle de GOLD) appliqué aux langues naturelles, qui constitue notre objectif.

C'est pourquoi il est nécessaire de s'abstraire autant que possible de ces contraintes, c'est-à-dire de rechercher des classes de langages plus étendues que les langages rigides et qui soient apprenables à partir de simples phrases plutôt qu'à partir de structures. Cela correspond exactement pour les grammaires AB à l'extension au cas des grammaires k-valuées apprises à partir de chaînes, démontrée par KANAZAWA dans [60] en passant par la propriété d'élasticité finie des langages rigides de structures (voir partie 3.3). Nous allons donc chercher des critères permettant d'obtenir l'élasticité finie des langages de structures[1] de certaines classes de GCG, de manière à pouvoir appliquer la méthode de KANAZAWA dans le but de trouver de telles classes apprenables "plus intéressantes".

Dans ce contexte, l'intérêt du modèle des GCG est double. D'abord, il permet de traiter uniformément de nombreux formalismes grammaticaux, y compris d'éventuels systèmes qui ne sont pas abordés dans cette étude : ainsi, le principe consiste simplement à fournir les critères qui rendent l'apprenabilité possible en fonction de la forme des règles ou de celle des grammaires, sans s'attacher à une instance particulière de $\mathcal{R}$-grammaires. Cela permet de réutiliser plus facilement ces résultats, sans devoir redémontrer l'apprenabilité pour chaque formalisme. Mais cela a aussi deux inconvénients : le formalisme en question doit être exprimable dans le modèle des GCG, et il est évidemment possible que certaines propriétés spécifiques d'un formalisme donné permettent d'obtenir l'apprenabilité de classes plus vastes que celles concernées par le résultat obtenu dans le cas général. Le second intérêt des GCG est beaucoup plus prosaïque : on a vu que la sous-classe des langages de structures rigides est apprenable, et on sait qu'il existe quelques sous-classes ayant l'élasticité finie. Nous essaierons donc de nous inspirer de ces cas particuliers pour découvrir les critères dont découle l'apprenabilité.

Dans ce chapitre nous traiterons d'un premier type de critères, inspirés des résultats de SHINOHARA obtenus à l'aide de la propriété de densité finie bornée (qui permet elle-même, sous certaines conditions, d'obtenir l'élasticité finie : voir partie 2.5). Techniquement, cela consiste à borner le nombre de grammaires réduites par rapport à un ensemble fini D d'éléments du langage. Pour cela on cherche en fait une borne sur la taille de ces grammaires : ceci a pour conséquence que nos critères imposent une contrainte peu pratique (car uniquement "technique"), à savoir une limite sur la taille des "types arguments". En revanche cette méthode autorise de nombreuses formes de règles universelles, autrement dit permet d'inclure une grande variété de formalismes grammaticaux.

Avant de donner ces critères et les démonstrations d'apprenabilité qui les accompagnent, nous présentons une petite extension de la définition de densité finie bornée donnée par SHINOHARA dans [95]. Notre définition étend la densité finie bornée à tout type de relation de réduction, contrairement à la version de SHINOHARA dans laquelle l'unique relation considérée est l'inclusion classique. Nous verrons dans le chapitre suivant que cela permet d'inclure des résultats tels que ceux de KANAZAWA pour les grammaires AB dans le cadre de la densité finie bornée, ce qui n'était pas le cas auparavant.

Remarque : le résultat d'apprenabilité des grammaires à arguments bornés, présenté dans ce chapitre, est une généralisation du cas des grammaires plates publié dans [78] : d'une part la limitation aux types à arguments plats (de taille 1) est assouplie aux types à arguments bornés par un entier k, et d'autre part

[1] Contrairement à KANAZAWA nous n'aurons pas besoin de la contrainte de rigidité dans ces démonstrations.

nous étendons ici le résultat aux grammaires contextuelles.

5.2 Généralisation de la densité finie bornée

Dans la partie 2.5, nous avons exposé la notion de densité finie bornée telle qu'elle est définie par SHINOHARA dans [95], ainsi que les conséquences importantes qui en découlent au niveau de l'apprenabilité des langages. On a vu que cette définition utilise la notion de grammaire réduite par rapport à un ensemble d'élements du langage, cette notion étant elle même définie à l'aide de la relation d'inclusion entre grammaires (voir partie 4.2). Or le fait de fixer l'inclusion simple comme unique relation de réduction entre deux grammaires exclut certaines formes de grammaires, même si celles-ci ont des propriétés très similaires à la densité finie bornée. C'est notamment le cas des grammaires AB, comme le montre l'exemple ci-dessous.

Exemple 5.1. Soit

- G_0 la grammaire $\{ x : s/a_0 \ ; \ y : a_0 \ ; \ z : s/b_0 \ ; \ w : b_0 \}$;
- σ_i la substitution définie par $\sigma_i = \{a_0 \mapsto A_i \ ; \ b_0 \mapsto b_i \}$ pour tout $i > 0$, avec $A_0 = b_i$ et $A_i = A_{i-1}/b_i$ pour tout $i > 0$;
- et $G_i = \sigma_i[G_0]$ pour tout $i > 0$.

Notons que pour tout $i > 0$ $G_i = \{ x : s/(((b_i/b_i)/\dots)/b_i) \ ; \ y : (((b_i/b_i)/\dots)/b_i) \ ; \ z : s/b_i \ ; \ w : b_i \}$.

Soit D l'ensemble de phrases $\{ xy \ ; \ zw \}$: il est clair que $D \subseteq L(G_0) \subseteq L(G_i)$ pour tout $i \geq 0$. De plus toutes les grammaires G_i ont le même nombre de règles et on voit bien qu'il est impossible de supprimer l'une de ces règles tout en conservant le fait que $D \subseteq L(G_i)$. Ainsi toute grammaire G_i est réduite par rapport à l'ensemble fini D.

Or $L(G_i) \neq L(G_j)$ pour tous $i \neq j$, car la chaîne zyw^i appartient à $L(G_j)$ si et seulement si $i = j$ (en effet le type $(((b_i/b_i)/\dots)/b_i)$ de y nécessite exactement i types b, et le seul moyen d'en produire exactement ce nombre est d'utiliser w). Ainsi il existe un nombre infini de grammaires réduites par rapport à un ensemble fini d'éléments du langage, dont les langages sont tous distincts, donc la classe des grammaires AB (même rigides) n'a pas la densité finie bornée.

Notons que cet exemple fonctionne également avec les langages de FA-structures.

Nous proposons dans cette partie une généralisation de la définition de densité finie bornée (ainsi que de ses conséquences) prenant en compte de tels cas. Il s'agit simplement de ne plus considérer la seule relation $\subseteq$ dans la définition de grammaire réduite mais toute relation de réduction vérifiant les critères appropriés. On parlera alors de *grammaire réduite selon la relation* $\preccurlyeq$ et de *densité finie bornée selon la relation* $\preccurlyeq$. Cette généralisation présente l'avantage de souligner les points communs entre les deux méthodes de démonstration d'apprenabilité de SHINOHARA et KANAZAWA. Nous en ferons usage notamment dans le chapitre 6.

5.2.1 Définitions

Définition 5.1 (Relation de réduction)**.** Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires, et $\preccurlyeq$ un ordre partiel[2] sur $\mathcal{G}$.

$\preccurlyeq$ est une *relation de réduction* pour $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ si les deux conditions suivantes sont remplies :

- Pour tout couple de grammaires $G, G' \in \mathcal{G}$ $G \preccurlyeq G'$ implique $\mathcal{M}(G) \subseteq \mathcal{M}(G')$.
- L'ordre $\prec$ est bien fondé.

Avec $\prec$ la relation définie sur $\mathcal{G}$ par : $G \prec G'$ si et seulement si $G \preccurlyeq G'$ et $G \neq G'$.

Remarque : D'un point de vue formel il serait souhaitable de définir la relation $\preccurlyeq$, et plus particulièrement l'égalité entre grammaires, sur des classes de grammaires plutôt que sur les grammaires, de manière à traiter de manière uniforme un ensemble de grammaires équivalentes entre elles (typiquement, l'ensemble des grammaires obtenues par un renommage des variables). Dans la mesure où la définition des classes d'équivalence des grammaires est spécifique au système de grammaire utilisé et afin de ne pas alourdir inutilement les notations, nous notons dans cette partie $G = G'$ le fait que les grammaires G et G' soient identiques *modulo une éventuelle relation d'équivalence* dans le système de grammaires concerné. En pratique, le renommage des variables sera le seul cas de grammaires équivalentes utilisé dans la suite.

Définition 5.2 (Grammaire $\preccurlyeq$-réduite)**.** Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires et $\preccurlyeq$ une relation de réduction pour $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$. Soit $D \subseteq \mathcal{U}$ un sous-ensemble fini de $\mathcal{U}$.

Une grammaire $G \in \mathcal{G}$ est *$\preccurlyeq$-réduite* par rapport à D si

- $D \subseteq \mathcal{M}(G)$
- Pour toute grammaire $G' \prec G$, $D \nsubseteq \mathcal{M}(G')$

Proposition 5.1. *Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires, D et D' deux sous-ensembles finis de $\mathcal{U}$ tels que $D \subseteq D'$, et $G \in \mathcal{G}$ une grammaire telle que $D \subseteq D' \subseteq \mathcal{M}(G)$.*

Si G est $\preccurlyeq$-réduite par rapport à D, alors G est aussi $\preccurlyeq$-réduite par rapport à D'.

Démonstration. G est $\preccurlyeq$-réduite par rapport à D, donc $D \nsubseteq \mathcal{M}(G')$ pour toute grammaire $G' \prec G$. Comme $D \subseteq D'$, on a aussi $D' \nsubseteq \mathcal{M}(G')$, donc G est $\preccurlyeq$-réduite par rapport à D'. □

Proposition 5.2. *Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires muni d'une relation de réduction $\preccurlyeq$, et $G \in \mathcal{G}$ une grammaire.*

Pour tout ensemble fini $D \subseteq \mathcal{M}(G)$ il existe une grammaire $G' \in \mathcal{G}$ telle que G' est $\preccurlyeq$-réduite par rapport à D et $G' \preccurlyeq G$.

Démonstration. Supposons qu'il existe un ensemble $D \subseteq \mathcal{M}(G)$ tel qu'il n'existe aucune grammaire $G' \preccurlyeq G$ $\preccurlyeq$-réduite par rapport à D. Pour tout $n \in \mathbb{N}$, on montre qu'il existe une séquence de grammaires $G_n \prec G_{n-1} \prec \cdots \prec G_1 \prec G$ telle que pour tout i $D \subseteq \mathcal{M}(G_i)$ mais G_i n'est pas $\preccurlyeq$-réduite par rapport à D :

- $n = 0$. $G_0 = G$. $G \preccurlyeq G$ donc par hypothèse G n'est pas $\preccurlyeq$-réduite par rapport à D.

[2] Rappelons qu'une relation $\preccurlyeq$ est un ordre partiel sur un ensemble E si $\preccurlyeq$ est réflexive ($a \preccurlyeq a$ pour tout $a \in E$), antisymétrique ($a \preccurlyeq b$ et $b \preccurlyeq a$ implique $a = b$) et transitive ($a \preccurlyeq b$ et $b \preccurlyeq c$ implique $a \preccurlyeq c$).

- $n > 0$. Par hypothèse d'induction $D \subseteq \mathcal{M}(G_{n-1})$ et G_{n-1} n'est pas $\preccurlyeq$-réduite par rapport à D, donc il existe une grammaire G_n telle que $G_n \prec G_{n-1}$ et $D \subseteq \mathcal{M}(G_n)$.

Ainsi une telle séquence existe pour tout $n \in \mathbb{N}$. Ceci implique que $\preccurlyeq$ n'est pas un ordre bien fondé sur $\mathcal{G}$, donc n'est pas une relation de réduction (définition 5.1). Par conséquent il existe une grammaire $G' \preccurlyeq$-réduite par rapport à D. □

Définition 5.3 ($MaxRed(\mathcal{G}, \preccurlyeq, k)$)**.** Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires et $\preccurlyeq$ une relation de réduction pour $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$.

Une grammaire $G \in \mathcal{G}$ appartient à $MaxRed(\mathcal{G}, \preccurlyeq, k)$ si et seulement si $n \leq k$ pour tout ensemble de grammaires $\{G_1, \ldots, G_n\} \subseteq \mathcal{G}$ tel que $G_1 \prec G_2 \prec \cdots \prec G_n \prec G$.

Remarque : Les propriétés suivantes sont des conséquences triviales de cette définition :

- $MaxRed(\mathcal{G}, \preccurlyeq, k) \subseteq \mathcal{G}$;
- Pour tout $k' \leq k$, $MaxRed(\mathcal{G}, \preccurlyeq, k') \subseteq MaxRed(\mathcal{G}, \preccurlyeq, k)$;
- Si $G \in MaxRed(\mathcal{G}, \preccurlyeq, k)$ et $G' \preccurlyeq G$, alors $G' \in MaxRed(\mathcal{G}, \preccurlyeq, k)$. En particulier, si $G \in MaxRed(\mathcal{G}, \preccurlyeq, k)$ et $G' \prec G$, alors $G' \in MaxRed(\mathcal{G}, \preccurlyeq, k-1)$.

Définition 5.4 (Densité finie bornée selon $\preccurlyeq$)**.** Un système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ muni d'une relation de réduction $\preccurlyeq$ a la *densité finie bornée selon* $\preccurlyeq$ si pour tout $k \geq 0$ et tout ensemble fini $D \subseteq \mathcal{U}$, l'ensemble

$$\{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}, \preccurlyeq, k) \text{ et } G \text{ est } \preccurlyeq\text{-réduite par rapport à } D \,\}$$

est fini.

Remarque : trivialement, si un système $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ a la densité finie bornée selon une relation $\preccurlyeq$, alors pour toute sous-classe $\mathcal{G}' \subseteq \mathcal{G}$ le système $\langle \mathcal{U}, \mathcal{G}', \mathcal{M} \rangle$ a aussi la densité finie bornée selon $\preccurlyeq$.

5.2.2 Apprenabilité des classes de densité finie bornée générale

Nous démontrons ici l'équivalent du théorème 2.3 donné par SHINOHARA dans [95], avec notre définition élargie de la densité finie bornée. Globalement cette démonstration suit le même principe que celle de SHINOHARA, excepté bien sûr que nous devons utiliser les propriétés de la relation de réduction (plus complexes car plus générales que dans le cas simple de l'inclusion).

Théorème 5.1. *Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires muni d'une relation de réduction $\preccurlyeq$.*

Si $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ a la densité finie bornée selon $\preccurlyeq$, alors

$$\mathcal{L}_k = \{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}, \preccurlyeq, k) \,\}$$

a l'élasticité finie, pour tout $k \in \mathbb{N}$.

Démonstration. On montre par induction sur k que $\mathcal{L}_k$ a l'élasticité finie.

$k = 0$. Soit $G \in \mathcal{G}$ et $D \subseteq \mathcal{M}(G)$ un ensemble fini. Si G n'est pas $\preccurlyeq$-réduite par rapport à D, alors il existe $G' \prec G$ telle que $D \subseteq \mathcal{M}(G')$, ce qui implique que $G \notin MaxRed(\mathcal{G}, \preccurlyeq, 0)$. Par conséquent toute grammaire $G \in MaxRed(\mathcal{G}, \preccurlyeq, 0)$ est $\preccurlyeq$-réduite par rapport à tout ensemble fini $D \subseteq \mathcal{M}(G)$, donc

$$\begin{aligned}\mathcal{L}_0 &= \{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}, \preccurlyeq, 0) \,\} \\ &\supseteq \{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}, \preccurlyeq, 0) \text{ et } D \subseteq \mathcal{M}(G) \,\} \\ &= \{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}, \preccurlyeq, 0) \text{ et } G \text{ est } \preccurlyeq\text{-réduite par rapport à } D \,\}.\end{aligned}$$

Or cet ensemble est fini car $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ a la densité finie bornée selon $\preccurlyeq$. Ainsi il ne peut exister aucune séquence infinie de langages différents dans $\mathcal{L}_0$, donc $\mathcal{L}_0$ a l'élasticité finie.

$k > 0$. On montre par contraposée que si $\mathcal{L}_{k-1}$ a l'élasticité finie, alors $\mathcal{L}_k$ a l'élasticité finie. On suppose donc que $\mathcal{L}_k$ a l'élasticité infinie. Comme pour tout $L \in \mathcal{L}_k$ il existe une grammaire $G \in MaxRed(\mathcal{G}, \preccurlyeq, k)$ telle que $\mathcal{M}(G) = L$, il existe une séquence infinie d'objets $\langle a_0, a_1, a_2, \ldots \rangle$ et une séquence infinie de grammaires $\langle G_1, G_2, \ldots \rangle$ telles que

$$\{a_0, \ldots, a_{i-1}\} \subseteq \mathcal{M}(G_i) \quad \text{et} \quad a_i \notin \mathcal{M}(G_i).$$

Pour tout $i > 0$ $\{a_0, \ldots, a_{i-1}\} \subseteq \mathcal{M}(G_i)$ et $G_i \in MaxRed(\mathcal{G}, \preccurlyeq, k)$, donc d'après la proposition 5.2 il existe une grammaire $G'_i \preccurlyeq G_i$ qui est $\preccurlyeq$-réduite par rapport à $\{a_0, \ldots, a_{i-1}\}$. Par définition $G'_i \preccurlyeq G_i$ implique $\mathcal{M}(G'_i) \subseteq \mathcal{M}(G_i)$, or $a_i \notin \mathcal{M}(G_i)$, donc $a_i \notin \mathcal{M}(G'_i)$. De plus $G'_i \in MaxRed(\mathcal{G}, \preccurlyeq, k)$, car $G'_i \preccurlyeq G_i \in MaxRed(\mathcal{G}, \preccurlyeq, k)$.

Soit h la fonction définie pour tout $i > 0$ par

$$h(i) = min(\{\, j \in \mathbb{N} \mid G'_i \text{ est } \preccurlyeq\text{-réduite par rapport à } \{a_0, \ldots, a_j\} \,\})$$

$h(i) < i$, car $a_i \notin G'_i$, donc G'_i ne peut pas être $\preccurlyeq$-réduite par rapport à un ensemble D contenant a_i.

On montre d'abord que l'ensemble $\{\, h(i) \mid i \in \mathbb{N}\}$ n'est pas borné : supposons au contraire qu'il le soit, c'est-à-dire qu'il existe une borne j_0 tel que $h(i) \leq j_0$ pour tout $i > 0$. Donc pour tout $i > j_0$, G'_i est $\preccurlyeq$-réduite par rapport à $\{a_0, \ldots, a_{j_0}\}$. $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ a la densité finie bornée selon $\preccurlyeq$, donc l'ensemble

$$\{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}, \preccurlyeq, k) \text{ et } G \text{ est } \preccurlyeq\text{-réduite par rapport à } \{a_0, \ldots, a_{j_0}\} \,\}$$

est fini. Or $\{\, \mathcal{M}(G'_i) \mid i > j_0 \,\}$ est infini (car $\forall j < i \; a_j \notin \mathcal{M}(G'_j)$ mais $a_j \in \mathcal{M}(G'_i)$), ce qui contredit l'hypothèse selon laquelle $\{\, h(i) \mid i \in \mathbb{N}\}$ est borné.

Par conséquent il existe une séquence infinie strictement croissante $i_1 < i_2 < \ldots$ telle que $h(i_j) < h(i_{j+1})$ pour tout $j > 0$. Soit $D_j = \{a_0, \ldots, a_{h(i_j)-1}, a_{h(i_j)}\}$ et $D'_j = \{a_0, \ldots, a_{h(i_j)-1}\}$, pour $j > 0$ quelconque. Par définition de h, G'_{i_j} n'est pas réduite par rapport à D'_j, sinon on aurait $h(i_j) \leq h(i_j) - 1$. Or $D'_j \subseteq \mathcal{M}(G'_{i_j})$, puisque $h(i_j) - 1 < i_j$, donc il existe une grammaire $G''_{i_j} \prec G'_{i_j}$ telle que $D'_j \subseteq \mathcal{M}(G''_{i_j})$. G'_{i_j} est $\preccurlyeq$-réduite par rapport à D_j (par définition de h) et $G''_{i_j} \prec G'_{i_j}$, donc $D_j \nsubseteq \mathcal{M}(G''_{i_j})$. Comme $D_j = D'_j \cup \{a_{h(i_j)}\}$ et $D'_j \subseteq \mathcal{M}(G''_{i_j})$, on a $a_{h(i_j)} \notin \mathcal{M}(G''_{i_j})$.

$\{a_0, a_{h(i_1)}, \ldots, a_{h(i_{j-1})}\} \subseteq \{a_0, a_1, \ldots, a_{h(i_j)-1}\}$, on obtient donc une séquence infinie d'objets $\langle a_0, a_{h(i_1)}, a_{h(i_2)} \ldots \rangle$ et une séquence infinie de grammaires $\langle G''_{i_1}, G''_{i_2}, \ldots \rangle$ telles que pour tout $j > 0$ $\{a_0, a_{h(i_1)}, \ldots, a_{h(i_j)-1}\} \subseteq \mathcal{M}(G_{i_j})$ et $a_{h(i_j)} \notin \mathcal{M}(G''_{i_j})$. Or $G''_{i_j} \prec G'_{i_j} \in MaxRed(\mathcal{G}, \preccurlyeq, k)$, donc $G''_{i_j} \in MaxRed(\mathcal{G}, \preccurlyeq, k-1)$ pour tout $j > 0$. Par conséquent ces deux séquences infinies démontrent l'élasticité infinie de $\mathcal{L}_{k-1}$. □

La définition ci-dessous présente la notion de grammaire faiblement $\preccurlyeq$-réduite. Contrairement à la définition 5.2 cette notion est indépendante d'un quelconque ensemble fini. Intuitivement on peut considérer qu'une grammaire est faiblement $\preccurlyeq$-réduite si elle ne contient "rien d'inutile" pour générer son langage.

Définition 5.5. Dans le système de grammaire $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ muni de la relation de réduction $\preccurlyeq$, une grammaire $G \in \mathcal{G}$ est *faiblement $\preccurlyeq$-réduite* si $\mathcal{M}(G) \neq \mathcal{M}(G')$ pour toute grammaire $G' \in \mathcal{G}$ telle que $G' \prec G$.

Il est clair, selon cette définition, que si une grammaire G est $\preccurlyeq$-réduite par rapport à un ensemble D, alors G est faiblement $\preccurlyeq$-réduite, ainsi cette version de la réduction d'une grammaire est plus faible que la précédente. Dans [60], KANAZAWA considère uniquement cette définition de la réduction des grammaires : ainsi ce que nous nommons ici "grammaire faiblement réduite" est appelée simplement "grammaire réduite" dans [60].

Proposition 5.3. *Pour toute grammaire G il existe une grammaire $G' \preccurlyeq G$ faiblement $\preccurlyeq$-réduite telle que $\mathcal{M}(G') = \mathcal{M}(G)$.*

Démonstration. Trivial par la définition 5.5. □

La proposition ci-dessous est destinée à simplifier l'utilisation du théorème 5.1, dont elle est une conséquence directe. Elle permet de négliger les variantes non faiblement réduites des grammaires d'une classe donnée, puisque qu'il existe toujours une grammaire faiblement réduite de même langage.

Proposition 5.4. *Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires et $\preccurlyeq$ une relation de réduction pour ce système. Soit $\mathcal{G}'$ l'ensemble des grammaires faiblement réduites de $\mathcal{G}$:*

$$\mathcal{G}' = \{\, G \in \mathcal{G} \mid G \text{ est faiblement } \preccurlyeq\text{-réduite} \,\}.$$

Si $\langle \mathcal{U}, \mathcal{G}', \mathcal{M} \rangle$ a la densité finie bornée selon $\preccurlyeq$ et s'il existe $k \in \mathbb{N}$ tel que $\mathcal{G}' \subseteq MaxRed(\mathcal{G}', \preccurlyeq, k)$, alors la classe de langages $\{\, \mathcal{M}(G) \mid G \in \mathcal{G} \,\}$ a l'élasticité finie.

Démonstration. Pour toute grammaire $G \in \mathcal{G}$ il existe une grammaire $G' \in \mathcal{G}'$ faiblement $\preccurlyeq$-réduite telle que $\mathcal{M}(G') = \mathcal{M}(G)$ (proposition 5.3), donc $\{\, \mathcal{M}(G) \mid G \in \mathcal{G} \,\} \subseteq \{\, \mathcal{M}(G) \mid G \in \mathcal{G}' \,\}$. Or il existe $k \geq 0$ tel que $\mathcal{G}' \subseteq MaxRed(\mathcal{G}', \preccurlyeq, k)$, donc $\{\, \mathcal{M}(G) \mid G \in \mathcal{G}' \,\} \subseteq \{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}', \preccurlyeq, k) \,\}$. $\langle \mathcal{U}, \mathcal{G}', \mathcal{M} \rangle$ a la densité finie bornée selon $\preccurlyeq$, donc selon le théorème 5.1 la classe de langages $\{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}', \preccurlyeq, k) \,\}$ a l'élasticité finie. Par conséquent la sous-classe $\{\, \mathcal{M}(G) \mid G \in \mathcal{G} \,\} \subseteq \{\, \mathcal{M}(G) \mid G \in MaxRed(\mathcal{G}', \preccurlyeq, k) \,\}$ a aussi l'élasticité finie. □

Proposition 5.5. *Soient $\mathcal{G}$, $\mathcal{G}'$ et $\mathcal{G}''$ des ensembles de grammaires tels que $\mathcal{G} \subseteq \mathcal{G}' \subseteq \mathcal{G}''$.*
Si $\mathcal{G} \subseteq MaxRed(\mathcal{G}'', \preccurlyeq, k)$ alors $\mathcal{G} \subseteq MaxRed(\mathcal{G}', \preccurlyeq, k)$.

Démonstration. Par définition de $MaxRed$: si toute séquence $G_1 \prec G_2 \prec \cdots \prec G_n \prec G$ vérifie $n \leq k$ lorsque $\{G_1, \ldots, G_n\} \subseteq \mathcal{G}''$, alors la propriété reste vraie lorsque $\{G_1, \ldots, G_n\} \subseteq \mathcal{G}'$ puisque $\mathcal{G}' \subseteq \mathcal{G}''$. Comme $\mathcal{G} \subseteq \mathcal{G}'$, $G \in \mathcal{G}'$ donc $\mathcal{G} \subseteq MaxRed(\mathcal{G}', \preccurlyeq, k)$. □

5.2.3 Exemple : les langages de motifs

Nous avons vu dans la partie 2.5 que la relation d'inclusion proposée par SHINOHARA dans la définition d'origine permet déjà d'obtenir des résultats d'apprenabilités impressionants (en particulier avec les grammaires contextuelles). Nous utiliserons la version généralisée de la densité finie bornée avec les grammmaires combinatoires générales dans le chapitre 6, en incluant au passage les grammaires AB. Avant cela, nous proposons, pour illustrer l'intérêt de cette généralisation, de revisiter le résultat d'ANGLUIN sur les *langages de motifs* (*pattern languages*) [4], qui a été chronologiquement la source du renouveau du modèle d'apprentissage de GOLD. Cet exemple est cependant très limité car on n'y retrouve pas l'intégralité du résultat d'ANGLUIN.

Définition 5.6 (Langages de motifs)**.** Un *motif* est une séquence non vide sur $\Sigma \cup X$, où Σ est le vocabulaire terminal et X un ensemble fini de variables. L'ensemble des motifs est noté P.

Soit $\leq'$ la relation définie sur P par $x \leq' y$ si et seulement s'il existe une substitution $\sigma : X \mapsto (X \cup \Sigma^+)$ telle que $x = \sigma(y)$.

Le langage d'un motif p est l'ensemble $L(p) = \{ x \in \Sigma^+ \mid x \leq' p \}$.

On considère que deux motifs ne différant que par un renommage de variables sont équivalents (on a alors $p \leq' p'$ et $p' \leq' p$).

Proposition 5.6. *$\leq'$ est une relation de réduction dans le système de grammaires $\langle \Sigma^+, P, L \rangle$.*

Démonstration. On constate aisément que $\leq'$ est un ordre partiel.

Soient $p, p' \in P$ tels que $p \leq' p'$, autrement dit il existe σ tel que $\sigma(p') = p$. Soit $x \in L(p)$: il existe τ telle que $\tau(p) = x$, donc il existe une substitution $\tau \circ \sigma$ telle que $(\tau \circ \sigma)(p') = x$. Par conséquent $x \in L(p')$, d'où $L(p) \subseteq L(p')$.

Montrons maintenant que $\leq'$ est bien fondé : supposons qu'au contraire il existe un ensemble de motifs $S \subseteq P$ tel que S n'a pas d'élément minimal, autrement dit pour tout motif $p \in S$ il existe un motif p' tel que $p' <' p$. Ainsi il existe une séquence infinie $\langle p_i \rangle_{i \in \mathbb{N}}$ telle que $p_i <' p$ et $p_i <' p_{i+1}$ pour tout $i \geq 0$. On note que $p <' p'$ si et seulement s'il existe σ telle que $\sigma(p') = p$ et σ n'est pas un renommage de variables. Cela implique soit qu'il existe une variable $v \in X$ telle que $\sigma(v) \in \Sigma^+$, soit qu'il existe deux variables $v, v' \in X$ telles que $\sigma(v) = \sigma(v')$. Dans les deux cas le nombre de variables distinctes présentes dans le motif diminue strictement lorsqu'on applique la substitution, il ne peut donc pas exister une telle séquence infinie. Par conséquent $\leq'$ est bien fondé. □

Proposition 5.7. *Le système de grammaires $\langle \Sigma^+, P, L \rangle$ a la densité finie bornée selon $\leq'$.*

Démonstration. Par définition, $p \leq' p'$ implique $|p'| \leq |p|$. Par conséquent le nombre de motifs p' tels que $p \in L(p')$ est fini, et a fortiori le nombre de motifs p' tels que p' est $\leq'$-réduit par rapport à p. Il est donc évident que $\{ L(p) \mid p \in MaxRed(P, \leq', k)$ et p est $\leq' -$réduit par rapport à $D \}$ est fini pour tout ensemble D et tout $k \geq 0$. □

Proposition 5.8. *Pour tout $k > 0$, la classe des langages de motifs*

$$\{\ L(p) \mid p \textit{ contient moins de } k \textit{ variables distinctes}\ \}$$

est apprenable.

Démonstration. On a vu dans la démonstration de la proposition 5.6 que le nombre de variables distinctes dans un motif p est une borne sur la taille de toute séquence $p'_1 <' p'_2 <' \cdots <' p$. Donc $\{\ L(p) \mid p$ contient moins de k variables distinctes $\} \subseteq MaxRed(P, \leq', k)$. Selon le théorème 5.1, cette classe a l'élasticité finie et est par conséquent apprenable. □

5.3 Apprenabilité des grammaires à arguments bornés

Nous proposons ici un premier résultat qui répond (en partie) à la question laissée ouverte par KANAZAWA concernant la caractérisation de sous-classes de grammaires combinatoires générales ayant l'élasticité finie. Il s'agit en fait d'une condition suffisante sur la forme des règles universelles permettant d'obtenir l'élasticité finie pour certaines familles de grammaires.

Pour trouver ce critère et prouver qu'il constitue une condition suffisante pour l'élasticité finie, nous avons utilisé ici l'approche de SHINOHARA [95]. Nous avons donc cherché quelles classes de grammaires, dans le modèle des GCG, ont la propriété de densité finie bornée selon la relation de réduction classique d'inclusion simple. La méthode est similaire à celle utilisée par SHINOHARA pour montrer la densité finie bornée des grammaires contextuelles. Elle consiste à exhiber une borne sur la taille des grammaires (dont le nombre de règles est borné) réduites par rapport à un ensemble d'éléments du langage, de sorte que l'ensemble des langages correspondant à ces grammaires est fini. Mais il y a une différence importante entre le cas des grammaires contextuelles et celui que nous étudions : la taille d'une grammaire contextuelle s'exprime en deux dimensions (le nombre de règles et le nombre de symboles/types[3] dans chaque règle), tandis que le modèle des GCG impose de tenir compte d'une troisième dimension (la taille de chaque type).

Le nombre de règles des grammaires considérées est déjà borné, par définition de la densité finie bornée. Nous étudions ici uniquement le cas des grammaires lexicalisées, donc le nombre de types par règle est également limité. La question centrale consiste par conséquent à borner la taille des types dans les grammaires. Le critère proposé devra donc permettre que la taille des types d'une grammaire réduite par rapport à un ensemble D donné ne dépende que de cet ensemble. Pour répondre à un tel critère, les règles universelles doivent "contrôler" la diminution globale de taille au cours d'un pas de dérivation : en effet, si la taille d'une séquence de types pouvait diminuer dans n'importe quelle proportion au cours de la dérivation, alors il n'y aurait aucun moyen de garantir que les types du lexique (qui sont à la base de la dérivation) sont de taille limitée.

Le critère que nous proposons reprend la distinction entre types *principaux* et types *arguments* qui existe dans les grammaires AB. En généralisant cette distinction et en imposant des conditions adéquates

[3] Les types étant tous atomiques dans ce formalisme.

sur ces deux sortes de types au niveau des règles universelles, nous montrons qu'il est possible de borner la taille des types d'une grammaire réduite. De ceci découle la densité finie bornée, puis l'élasticité finie et finalement l'apprenabilité des classes de langages étudiées.

La première version de ces travaux a été présentée dans [78]. Nous en présentons ici une version étendue : le résultat principal est élargi aux grammaires contextuelles et aux grammaires à arguments bornés (c'est-à-dire pas seulement aux grammaires dites *plates*, qui en sont un cas particulier).

5.3.1 Grammaires combinatoires générales lexicalisées à argument bornés

Définition 5.7 (Type à arguments bornés)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs, et pour tout $f \in \mathcal{O}$ soit $arg_f : \{1, \ldots, \textit{arité}(f)\} \mapsto \{0, 1\}$ une fonction qui associe à chaque position dans f le statut d'argument (1) ou de non argument (0). Étant donné un ensemble de variables $\mathcal{V}$ et un entier $k > 0$, l'ensemble des *types à arguments borné par* k, noté $ArgB(\mathcal{V}, k)$, est défini comme le plus petit ensemble tel que

- $\mathcal{V} \subseteq ArgB(\mathcal{V}, k)$
- Pour tout $f \in \mathcal{O}$ avec $\textit{arité}(f) = n$, si $t_1, \ldots, t_n$ appartiennent à $ArgB(\mathcal{V}, k)$ et pour tout i $(1 \leq i \leq n)$ $arg_f(i) = 1 \Rightarrow \|t_i\| \leq k$, alors $f(t_1, \ldots, t_n) \in ArgB(\mathcal{V}, k)$.

Cette définition restreint les types au sous-ensemble des types dont chaque sous-type présent dans une position telle que $arg_f(i) = 1$ (position argument, voir ci-dessous) est de taille bornée par k. Malgré la similitude, il ne faut pas confondre notre définition des arguments bornés avec la notion plus classique d'*ordre des types* (ou grammaires) : un type est d'ordre k si la hauteur de ses sous-types arguments est toujours bornée par k. Ainsi on voit bien que même si un type à arguments bornés par k est effectivement toujours un type d'ordre k, la réciproque est fausse en général.

Remarque : Si un type t appartient à $ArgB(\mathcal{V}, k)$, alors tout sous-type de t appartient aussi à $ArgB(\mathcal{V}, k)$. En revanche, la réciproque est fausse en général.

Un sous-type u d'un type t est dit *en position argument dans* t s'il existe un sous-type w dans t tel que $w = f(t_1, \ldots, t_{i-1}, u, t_{i+1}, \ldots, t_n)$ et $arg_f(i) = 1$. Il faut noter ici qu'il suffit qu'une seule occurrence de u vérifie cette condition pour que u soit en position argument : il est donc possible qu'il y ait dans t une autre occurrence de u ne vérifiant pas la condition. Par définition, tous les sous-types d'un type $t \in ArgB(\mathcal{V}, k)$ qui sont en position argument sont de taille inférieure ou égale à k.

Définition 5.8 (Grammaire à arguments bornés)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire lexicalisée. G est une *grammaire à arguments bornés par* k si tout type $t \in Lex(G)$ appartient à $ArgB(Pr, k)$.

Dans le cas particulier où $k = 1$, un type $t \in ArgB(\mathcal{V}, 1)$ est dit *plat*. De même, une grammaire à arguments bornés par $k = 1$ est une *grammaire plate*[4]. Le fait que les arguments doivent avoir une taille d'au plus 1 signifie que tous les arguments sont des variables (ou des types primitifs dans le cas de la grammaire).

[4] Nous n'étudions l'apprenabilité que pour ce cas particulier dans l'article d'origine [78].

Définition 5.9 (Règle universelle à diminution limitée)**.** Soit R une règle universelle. $R = A_1 \dots A_n \rightarrow B_1 \dots B_m$ est une *règle universelle à diminution limitée* si les conditions suivantes sont remplies :

- R est contextuelle (voir définition 4.19),
- $\{A_1, \dots, A_n\} \cup \{B_1, \dots, B_m\} \subseteq ArgB(Var(R), 1)$,
- Pour toute variable $v \in Var(R)$, s'il n'existe aucun A_i ($1 \leq i \leq n$) tel que v soit en position argument dans A_i, alors quel que soit j ($1 \leq j \leq m$) v n'est pas en position argument dans B_j et

$$\sum_{i=1}^{n} \#_v(A_i) = \sum_{j=1}^{m} \#_v(B_j).$$

Remarque : Les deux premières conditions du critère proposé sont simples et peu contraignantes. En revanche la dernière condition est plus complexe : nous verrons son intérêt ci-dessous, mais il est utile de noter que la contrainte qu'elle impose porte seulement sur les types qui *ne sont pas* en position argument.

Exemple 5.2. En fixant les positions arguments de la façon habituelle indiquée ci-dessous,

$$arg_{/}(1) = arg_{\backslash}(2) = 0$$
$$arg_{/}(2) = arg_{\backslash}(1) = 1$$

les règles universelles des grammaires AB vérifient les conditions des règles à diminution limitée :

$$A/B\ B \ \rightarrow \ A$$
$$B\ B\backslash A \ \rightarrow \ A$$

En effet, dans les deux règles la variable B est le seul sous-type en position argument (dans A/B ou dans $B\backslash A$), donc tous les sous-types en position argument sont bien de taille 1. La seule variable qui n'est pas en position argument dans la partie gauche de la règle est A, et on peut constater que d'une part A n'est pas non plus en position argument à droite, et d'autre part le nombre d'occurrences de A à gauche et à droite est identique.

Il faut cependant noter que, dans la définition générale, tous les types des grammaires AB ne sont pas nécessairement à arguments bornés par k[5] : par exemple, le type $a/((b/c)/d)$ appartient à $ArgB(Pr, 5)$ mais pas à $ArgB(Pr, 4)$, car $\|(b/c)/d\| = 5$.

La proposition 5.9 ci-dessous montre la fermeture de la propriété, pour une séquence de types, d'avoir des arguments bornés par k est fermée par l'opération de dérivation par des règles à diminution limitée.

Proposition 5.9. *Soit R une règle à diminution limitée, Pr un ensemble fini de variables et $t_1, \dots, t_n$ des types à arguments bornés par k sur Pr.*

Si $t_1 \dots t_n \rightarrow_R u_1 \dots u_m$, alors $u_i \in ArgB(Pr, k)$ pour tout i, $1 \leq i \leq m$.

[5] k étant un entier fixé, indépendant du type. Autrement, il est clair que pour tout type t il existe un k suffisamment grand pour que $t \in ArgB(Pr, k)$.

Démonstration. Soit $R = A_1 \dots A_n \rightarrow B_1 \dots B_m$. Par définition, si $t_1 \dots t_n \rightarrow_R u_1 \dots u_m$ alors il existe une substitution σ telle que $\sigma(A_i) = t_i$ pour tout $1 \leq i \leq n$ et $\sigma(B_i) = u_i$ pour tout $1 \leq i \leq m$.

Supposons qu'il existe un j tel que $u_j = \sigma(B_j) \notin ArgB(Pr, k)$. Il existe donc un sous-type w dans $\sigma(B_j)$ tel que $w = f(w_1, \dots, w_l)$, ainsi qu'un indice p, $1 \leq p \leq l$, tel que $arg_f(p) = 1$ et $\|w_p\| > k$. R est une règle contextuelle, donc pour toute variable $v \in Var(R)$ le nombre d'occurrences de v dans la partie gauche est supérieur ou égal au nombre d'occurrences de v dans la partie droite. Par conséquent toute variable $v \in Var(R)$ (utilisée dans la règle) apparaît dans la partie gauche dans au moins un A_i. Or $t_i \in ArgB(Pr, k)$, donc comme $\sigma(v)$ est un sous-type de t_i, $\sigma(v)$ appartient aussi à $t_i \in ArgB(Pr, k)$. C'est pourquoi il n'existe aucun $v \in Var(R)$ tel que w soit un sous-type de $\sigma(v)$.

Ainsi w est un sous-type de $\sigma(B_j)$ mais il n'y a aucune variable v dans B_j tel que w soit un sous-type de $\sigma(v)$. Par conséquent il existe un sous-type $f(b_1, \dots, b_l)$ dans B_j tel que $\sigma(b_p) = w_p$. $B_j \in ArgB(Pr, 1)$ (car R est une règle à diminution limitée) et $arg_f(p) = 1$, donc $\|b_p\| = 1$, autrement dit $b_p \in Var(R)$. b_p est une variable en position argument dans la partie droite de R, donc d'après la définition 5.9 il existe au moins une occurrence de b_k en position argument dans la partie gauche : soit A_i tel que b_p soit en position argument dans A_i. $\sigma(A_i)$ contient $\sigma(b_p) = w_p$ en position argument et $\|w_p\| > k$, on obtient donc une contradiction : $\sigma(A_i) = t_i \notin ArgB(Pr, k)$.

□

La démonstration de la proposition précédente explique l'intérêt de la troisième condition du critère de diminution limitée des règles universelles. Celle-ci impose en effet aux types non arguments d'une part de ne pas devenir un type argument après un pas de dérivation, d'autre part de maintenir leur nombre global strictement identique avant et après un pas de dérivation. On voit que la première partie de la condition garantit qu'aucun type argument de taille quelconque ne puisse apparaître, ce qui invaliderait le fait que tous les types sont à arguments bornés. Comme on le verra dans la suite, ceci sera la base du contrôle des règles universelles sur la taille des types dérivés : globalement, les types arguments peuvent diminuer/disparaître mais on connaît leur taille maximale par avance, tandis que les types principaux (les autres) sont contraints de rester de taille identique.

5.3.2 Apprenabilité des grammaires à arguments bornés

Comme dans la preuve donnée par SHINOHARA qui démontre la densité finie bornée des grammaires contextuelles, nous montrons que pour tout ensemble fini D il existe une borne sur la taille des grammaires (à arguments bornés) réduites par rapport à D. Il s'agit donc ici de limiter la taille des types du lexique, en utilisant la forme particulière des règles à diminution limitée.

Proposition 5.10. *Soit R une règle à diminution limitée, Pr un ensemble fini de variables et $t_1, \dots, t_n$ des types à arguments bornés par k sur Pr.*

Si $t_1 \dots t_n \rightarrow_R u_1 \dots u_m$, alors

$$\sum_{i=1}^{n} \|t_i\| \leq \sum_{j=1}^{m} \|u_j\| + M_{(k,R)}$$

où $M_{(k,R)}$ est une constante qui ne dépend que de R et k.

Démonstration. Soit $R = A_1 \dots A_n \rightarrow B_1 \dots B_m$. Par définition $t_1 \dots t_n \rightarrow_R u_1 \dots u_m$ implique qu'il existe une substitution σ telle que $\sigma(A_i) = t_i$ (resp. $\sigma(B_j) = u_j$) pour tout i, $1 \leq i \leq n$ (resp. $1 \leq j \leq m$). Soit $C \in \{A_1, \dots A_n, B_1, \dots B_m\}$. On peut décomposer la taille de $\sigma(C)$ de la manière suivante :

$$\sigma(C) = \|C\| + \sum_{v \in Var(R)} (\#_v(C) \times (\|\sigma(v)\| - 1))$$

Soit $\{\mathcal{V}_{arg}, \mathcal{V}_{pr}\}$ la partition de $Var(R)$ définie par

$$\mathcal{V}_{arg} = \{\ v \in Var(R) \mid \text{il existe } A_i \text{ tel que } v \text{ est en position argument dans } A_i\ \}$$

et $\mathcal{V}_{pr} = Var(R) \setminus \mathcal{V}_{arg}$. Soient

$$x_{pr} = \sum_{i=1}^{n} \sum_{v \in \mathcal{V}_{pr}} (\#_v(A_i) \times (\|\sigma(v)\| - 1)), \quad x_{arg} = \sum_{i=1}^{n} \sum_{v \in \mathcal{V}_{arg}} (\#_v(A_i) \times (\|\sigma(v)\| - 1)),$$
$$y_{pr} = \sum_{j=1}^{m} \sum_{v \in \mathcal{V}_{pr}} (\#_v(B_j) \times (\|\sigma(v)\| - 1)), \quad y_{arg} = \sum_{j=1}^{m} \sum_{v \in \mathcal{V}_{arg}} (\#_v(B_j) \times (\|\sigma(v)\| - 1)).$$

Ainsi

$$\begin{aligned}
\sum_{i=1}^{n} \|t_i\| &= \sum_{i=1}^{n} \|\sigma(A_i)\| \\
&= \sum_{i=1}^{n} (\|A_i\| + \sum_{v \in V_{pr}} (\#_v(A_i) \times (\|\sigma(v)\| - 1)) + \sum_{v \in V_{arg}} (\#_v(A_i) \times (\|\sigma(v)\| - 1))) \\
&= \sum_{i=1}^{n} \|A_i\| + \sum_{i=1}^{n} \sum_{v \in V_{pr}} (\#_v(A_i) \times (\|\sigma(v)\| - 1)) + \sum_{i=1}^{n} \sum_{v \in V_{arg}} (\#_v(A_i) \times (\|\sigma(v)\| - 1)) \\
&= \sum_{i=1}^{n} \|A_i\| + x_{pr} + x_{arg}
\end{aligned}$$

Or R est une règle à diminution limitée, donc pour toute variable $v \in \mathcal{V}_{pr}$ on a

$$\sum_{i=1}^{n} \#_v(A_i) = \sum_{j=1}^{m} \#_v(B_j),$$

ce qui implique $\sum_{v \in \mathcal{V}_{pr}} \sum_{i=1}^{n} (\#_v(A_i) \times (\|\sigma(v)\| - 1)) = \sum_{v \in \mathcal{V}_{pr}} \sum_{j=1}^{m} (\#_v(B_j) \times (\|\sigma(v)\| - 1))$, c'est-à-dire $x_{pr} = y_{pr}$.

De plus tout t_i appartient à $ArgB(Pr, k)$, et par définition toute variable v appartenant à $\mathcal{V}_{arg}$ est en position argument dans un A_i, donc $\|\sigma(v)\| \leq k$ pour tout $v \in \mathcal{V}_{arg}$:

$$\begin{aligned}
x_{arg} - y_{arg} &= \sum_{i=1}^{n} \sum_{v \in \mathcal{V}_{arg}} (\#_v(A_i) \times (\|\sigma(v)\| - 1)) - \sum_{j=1}^{m} \sum_{v \in \mathcal{V}_{arg}} (\#_v(B_j) \times (\|\sigma(v)\| - 1)) \\
&= \sum_{v \in \mathcal{V}_{arg}} (\ (\sum_{i=1}^{n} \#_v(A_i) - \sum_{j=1}^{m} \#_v(B_j)) \ \times \ (\|\sigma(v)\| - 1)\ ) \\
&\leq \sum_{v \in \mathcal{V}_{arg}} (\ (\sum_{i=1}^{n} \#_v(A_i) - \sum_{j=1}^{m} \#_v(B_j)) \ \times \ (k - 1)\ )
\end{aligned}$$

Comme $\sum_{j=1}^{m} \|u_j\| = \sum_{j=1}^{m} \|B_j\| + y_{pr} + y_{arg}$, on obtient :

$$\begin{aligned}\sum_{i=1}^{n} \|t_i\| &= \sum_{j=1}^{m} \|u_j\| + \sum_{i=1}^{n} \|A_i\| + x_{pr} + x_{arg} - \sum_{j=1}^{m} \|B_j\| - y_{pr} - y_{arg} \\ &= \sum_{j=1}^{m} \|u_j\| + \sum_{i=1}^{n} \|A_i\| - \sum_{j=1}^{m} \|B_j\| + x_{arg} - y_{arg} \\ &\leq \sum_{j=1}^{m} \|u_j\| + \sum_{i=1}^{n} \|A_i\| - \sum_{j=1}^{m} \|B_j\| + \sum_{v \in \mathcal{V}_{arg}} ((\sum_{i=1}^{n} \#_v(A_i) - \sum_{j=1}^{m} \#_v(B_j)) \times (k-1))\end{aligned}$$

Soit

$$M_{(k,R)} = \sum_{i=1}^{n} \|A_i\| - \sum_{j=1}^{m} \|B_j\| + \sum_{v \in \mathcal{V}_{arg}} \left(\left(\sum_{i=1}^{n} \#_v(A_i) - \sum_{j=1}^{m} \#_v(B_j) \right) \times (k-1) \right),$$

cette valeur ne dépend bien que de la règle R et de la constante k. □

Proposition 5.11. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire lexicalisée à arguments bornés par k, S une $\mathcal{R}$-structure et P une structure de dérivation pour G ayant pour entrée la séquence $\langle t_1, \ldots, t_n \rangle$ et pour sortie $\langle u_1, \ldots, u_m \rangle$.*

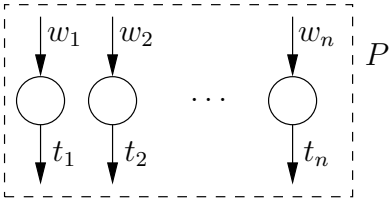
Si P est une instance de S, alors

$$\sum_{i=1}^{n} \|t_i\| \leq M_{(k,S,\mathcal{R})} + \sum_{j=1}^{m} \|u_j\|$$

où $M_{(k,S,\mathcal{R})}$ est une constante qui ne dépend que de k, de la structure S et de l'ensemble de règles universelles $\mathcal{R}$.

Démonstration. Soit $M_{\mathcal{R}} = max(\{ M_{(k,R)} \mid R \in \mathcal{R} \})$, $M_{(k,R)}$ étant la constante utilisée dans la proposition 5.10, et soit $M_{(k,S,\mathcal{R})} = N \times M_{\mathcal{R}}$, N étant le nombre de nœuds étiquetés par le nom d'une règle universelle[6]) dans S. Rappelons que si P est une instance de S les deux structures sont isomorphes, donc le nombre de nœuds dans S et dans P est identique. On montre par induction sur le nombre de nœuds N dans la $\mathcal{R}$-structure S que $\{u_1, \ldots, u_m\} \subseteq ArgB(Pr, k)$ et que $\sum_{i=1}^{n} \|t_i\| \leq N \times M_{\mathcal{R}} + \sum_{j=1}^{m} \|u_j\|$ pour tout $N \geq 0$.

$N = 0$. P est de la forme



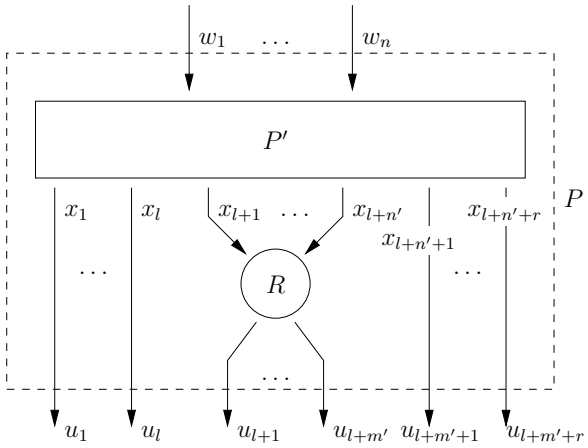
[6]Remarque : la grammaire G est lexicalisée, on sait donc qu'il y a exactement n nœuds dans S correspondant aux applications des règles locales sur les n entrées. Par conséquent N est égal au nombre total de nœuds dans S moins n.

puisque G est lexicalisée (voir partie 4.2.1.1), et on a $n = m$ et $t_i = u_i$ pour tout i. Clairement si $\sum_{i=1}^{n} \|t_i\| = \sum_{j=1}^{m} \|u_j\|$ on a $\sum_{i=1}^{n} \|t_i\| \leq N \times M_{\mathcal{R}} + \sum_{j=1}^{m} \|u_j\|$. De plus $t_i = u_i \in ArgB(Pr, k)$ car G est à arguments bornés par k.

$N > 0$. On suppose que la propriété est vraie pour toute $\mathcal{R}$-structure de $N-1$ nœuds. Comme G est lexicalisée, P est nécessairement constituée de l'application d'une règle universelle R sur une structure P' ayant $N-1$ nœuds étiquetés par une règle universelle. Soit

$$\langle x_1, \ldots, x_l, x_{l+1}, \ldots, x_{l+n'}, x_{l+n'+1}, \ldots, x_{l+n'+r} \rangle$$

la sortie de P', dans laquelle la sous-séquence $\langle x_{l+1}, \ldots, x_{l+n'} \rangle$ est l'entrée du nœud étiqueté R, et soit $\langle u_1, \ldots, u_l, u_{l+1}, \ldots, u_{l+m'}, u_{l+m'+1}, \ldots, u_{l+m'+r} \rangle$ (avec $l+m'+r = m$) la sortie de P, dans laquelle la sous-séquence $\langle u_{l+1}, \ldots, u_{l+m'} \rangle$ est la sortie du nœud de la règle R. P est donc de la forme suivante :



Par hypothèse d'induction, les sorties de S' $x_1, \ldots, x_{l+n'+r}$ appartiennent à $ArgB(Pr, k)$.
$x_{l+1} \ldots x_{l+n'} \rightarrow_R u_{l+1} \ldots u_{l+m'}$, donc d'après la proposition 5.9 $\{u_{l+1}, \ldots, u_{l+m'}\} \subseteq ArgB(Pr, k)$. Par conséquent la proposition 5.10 donne

$$\sum_{i=1}^{n'} \|x_{l+i}\| \leq \sum_{j=1}^{m'} \|u_{l+j}\| + M_{(k,R)}.$$

$u_i = x_i$ pour tout $1 \leq i \leq l$ et $x_{l+n'+i} = u_{l+m'+i}$ pour tout $1 \leq i \leq r$, donc

$$\begin{aligned}
\sum_{i=1}^{l+n'+r} \|x_i\| &= \sum_{i=1}^{l} \|x_i\| + \sum_{i=1}^{n'} \|x_{l+i}\| + \sum_{i=1}^{r} \|x_{l+n'+i}\| \\
&= \sum_{i=1}^{l} \|u_i\| + \sum_{i=1}^{n'} \|x_{l+i}\| + \sum_{i=1}^{r} \|u_{l+m'+i}\| \\
&\leq \sum_{i=1}^{l} \|u_i\| + \sum_{j=1}^{m'} \|u_{l+j}\| + M_{(k,R)} + \sum_{i=1}^{r} \|u_{l+m'+i}\| \\
&\leq \sum_{j=1}^{l+m'+r} \|u_j\| + M_{(k,R)}
\end{aligned}$$

La structure S' a $N-1$ nœuds, donc par hypothèse d'induction

$$\begin{aligned}\sum_{i=1}^{n}\|t_i\| &\leq (N-1)\times M_{\mathcal{R}} + \sum_{i=1}^{l+n'+r}\|x_i\| \\ &\leq (N-1)\times M_{\mathcal{R}} + \sum_{j=1}^{l+m'+r}\|u_j\| + M_{(k,R)}\end{aligned}$$

$\langle u_1,\ldots,u_l,u_{l+1},\ldots,u_{l+m'},u_{l+m'+1},\ldots,u_{l+m'+r}\rangle = \langle u_1,\ldots,u_m\rangle$, et par définition $M_{(k,R)} \leq M_{\mathcal{R}} = max(\{\, M_{(k,R)} \mid R\in\mathcal{R}\,\})$, donc

$$\sum_{i=1}^{n}\|t_i\| \leq N\times M_{\mathcal{R}} + \sum_{j=1}^{m}\|u_j\|$$

□

Comme nous utiliserons la relation classique d'inclusion $\subseteq$ comme relation de réduction pour la densité finie bornée, nous montrons ci-dessous que celle-ci répond bien aux critères d'une telle relation.

Soit $\mathcal{R}$ un ensemble de règles universelles et $\mathcal{G}_{\mathcal{R}}$ l'ensemble des $\mathcal{R}$-grammaires. $\subseteq$ (voir partie 4.2) est une relation de réduction (voir définition 5.1) pour le système de grammaires $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}_{\mathcal{R}}, SL\rangle$:

- $\forall G,G'\in\mathcal{G}_{\mathcal{R}}, G\subseteq G' \Rightarrow SL(G)\subseteq SL(G')$. En effet, si G' contient toutes les règles locales de G alors toutes les structures de dérivation pour G sont valides pour G'. En particulier toute structure de dérivation de G dont la sortie est $\langle s\rangle$ et qui est une instance d'une $\mathcal{R}$-structure S est valide pour G', ce qui implique par définition que S appartient à G'.
- $\subseteq$ est clairement un ordre bien fondé.

Proposition 5.12. *Pour toute classe $\mathcal{G}$ de $\mathcal{R}$-grammaires et tout $k\in\mathbb{N}$:*

$$MaxRed(\mathcal{G},\subseteq,k) = \{\, G\in\mathcal{G} \mid |G|\leq k\,\}$$

Démonstration. $MaxRed(\mathcal{G},\subseteq,k) \subseteq \{\, G\in\mathcal{G}\mid |G|\leq k\,\}$. Supposons qu'il existe une grammaire $G\in MaxRed(\mathcal{G},\subseteq,k)$ telle que $|G|>k$: soit G_n la grammaire obtenue en retirant arbitrairement une seule règle dans G, puis G_{n-1} la grammaire obtenue en retirant une seule autre règle dans G_n, et ainsi de suite jusqu'à G_1 qui ne contient aucune règle : de toute évidence $n=|G|$. On a donc construit une séquence $G_1\subset G_2\subset\cdots\subset G_n\subset G$ avec $n>k$, ce qui contredit le fait que G appartienne à $MaxRed(\mathcal{G},\subseteq,k)$.

$MaxRed(\mathcal{G},\subseteq,k) \supseteq \{\, G\in\mathcal{G}\mid |G|\leq k\,\}$. $G_i\subset G_{i+1}$ implique que $|G_i|<|G_{i+1}|$, donc comme $|G|\geq 0$ on a $n\leq k$ pour toute séquence $G_1\subset G_2\subset\cdots\subset G_n\subset G$. □

Proposition 5.13. *Soit D un ensemble fini de $\mathcal{R}$-structures et G une grammaire lexicalisée à arguments bornés par k.*

Si G est $\subseteq$-réduite par rapport à D alors tout type $t\in Lex(G)$ vérifie

$$\|t\| \leq M_{(k,D,\mathcal{R})} + \|s\|$$

où $M_{(k,D,\mathcal{R})}$ est une constante qui ne dépend que de k, de l'ensemble fini D et de l'ensemble de règles universelles $\mathcal{R}$.

Démonstration. Pour toute $\mathcal{R}$-structure $S \in D$, Soit E_S l'ensemble des structures de dérivations P pour G telles que P est une instance de S et la sortie de P est $\langle s \rangle$. $D \subseteq SL(G)$, donc pour tout $S \in D$ l'ensemble E_S n'est pas vide (définition 4.8). Pour toute règle locale $w \triangleright t$ dans la grammaire G, on montre d'abord qu'il existe une $\mathcal{R}$-structure S dans l'ensemble D et une structure de dérivation $P \in E_S$ telles que l'entrée de P contient t. Supposons que ce ne soit pas le cas, c'est-à-dire que pour toute structure $S \in D$ il n'existe aucun $P \in E_S$ dont l'entrée contient t. Soit G' la grammaire constituée de toute les règles de G sauf $w \triangleright G$: pour tout $S \in D$ il existe une instance de S pour G' dont la sortie est $\langle s \rangle$, puisqu'il en existe une pour G dans E_S et que celle-ci n'utilise pas la règle $w \triangleright t$. Par conséquent $D \subseteq SL(G')$, et comme $G' \subset G$ cela contredit l'hypothèse selon laquelle G est $\subseteq$-réduite par rapport à D.

Soit $M_{(k,D,\mathcal{R})} = max(\{ M_{(k,S,\mathcal{R})} \mid S \in D \})$, où $M_{(k,S,\mathcal{R})}$ est la constante définie pour une structure S dans la proposition 5.11. Pour toute règle $w \triangleright t$ de G il existe $P \in E_S$ dont l'entrée $\langle t_1, \ldots, t_n \rangle$ contient t, donc d'après la proposition 5.11 :

$$\|t\| \;\leq\; \sum_{i=1}^{n} \|t_i\| \;\leq\; M_{(k,S,\mathcal{R})} + \|s\| \;\leq\; M_{(k,D,\mathcal{R})} + \|s\|$$

□

Théorème 5.2. *Soit k un entier positif, $\mathcal{R}$ un ensemble de règles universelles à diminution limitée et $\mathcal{G}_{(k,\mathcal{R})}$ l'ensemble des $\mathcal{R}$-grammaires lexicalisées à arguments bornés par k.*

Le système de grammaires $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}_{(k,\mathcal{R})}, SL \rangle$ a la densité finie bornée selon $\subseteq$.

Démonstration. Soi k' un entier positif et $D \subseteq SL(\mathcal{R})_\Sigma$ un ensemble fini de $\mathcal{R}$-structures. Soit $G \in \mathcal{G}_{(k,\mathcal{R})}$ une grammaire $\subseteq$-réduite par rapport à D telle que $G \in MaxRed(\mathcal{G}_{(k,\mathcal{R})}, \subseteq, k')$. D'après la proposition 5.13, tout type $t \in Lex(G)$ est de taille inférieure ou égale à $M_{(k,D,\mathcal{R})} + \|s\|$ (avec $M_{(k,D,\mathcal{R})}$ la constante définie dans la proposition 5.13). De plus, la proposition 5.12 donne $|G| \leq k'$. Ainsi la grammaire G ne peut contenir plus de k' règles, chaque règle étant de la forme $w \triangleright t$ avec $w \in \Sigma$ et t un type de taille bornée. Par conséquent le nombre de telles grammaires est borné (si l'on excepte les variantes par renommage des variables, qui sont équivalentes), donc le nombre des langages qu'elles représentent l'est aussi : pour tout $k' \geq 0$ et tout ensemble fini $D \subseteq SL(\mathcal{R})_\Sigma$, l'ensemble

$$\{\, SL(G) \mid G \in MaxRed(\mathcal{G}_{(k,\mathcal{R})}, \subseteq, k') \text{ et } G \text{ est } \subseteq\text{-réduite par rapport à } D \,\}$$

est fini.

□

Remarque : Contrairement au cas particulier des grammaires AB rigides dans lequel toute FA-structure est compatible avec au moins une grammaire, dans le cas général il est possible qu'aucune grammaire ne soit compatible avec un ensemble fini de $\mathcal{R}$-structures D de $SL(\mathcal{R})_\Sigma$. C'est pourquoi il n'existe pas nécessairement de grammaire réduite par rapport à D pour tout ensemble fini D, mais cela n'a aucune incidence sur le résultat précédent puisque l'ensemble vide (des grammaires réduites) est bien entendu fini.

Grâce au théorème de SHINOHARA, on obtient aisément le résultat suivant :

Corollaire 5.1. *Soit k un entier positif, $\mathcal{R}$ un ensemble de règles universelles à diminution limitée et $\mathcal{G}_{(k,\mathcal{R})}$ l'ensemble des $\mathcal{R}$-grammaires lexicalisées à arguments bornés par k. Pour tout $k' \geq 0$, la classe des langages de structures*

$$\{\ SL(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})} \text{ et } G \text{ est } k'\text{-valuée}\ \}$$

a l'élasticité finie pour tout $k' \geq 0$.

Démonstration. La simple application du théorème 5.1 au résultat du théorème 5.2 indique que la classe $\{\ SL(G) \mid G \in MaxRed(\mathcal{G}_{(k,\mathcal{R})}, \subseteq, k'')\ \}$ a l'élasticité finie pour tout $k'' \geq 0$. On sait que $MaxRed(\mathcal{G}_{(k,\mathcal{R})}, \subseteq, k'') = \{\ G \in \mathcal{G}_{(k,\mathcal{R})} \mid |G| \leq k''\ \}$ (proposition 5.12), or d'après la proposition 4.4 pour tout $k' \geq 0$ il existe $k'' \geq 0$ tel que $\{\, G \in \mathcal{G}_{(k,\mathcal{R})} \mid G \text{ est } k'\text{-valuée}\,\} \subseteq \{\, G \in \mathcal{G}_{(k,\mathcal{R})} \mid |G| \leq k''\,\}$. Par conséquent la classe $\{\ SL(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})}$ et G est k'-valuée $\}$ a l'élasticité finie pour tout $k' \geq 0$. □

Corollaire 5.2. *Soit k un entier positif, $\mathcal{R}$ un ensemble de règles universelles à diminution limitée et $\mathcal{G}_{(k,\mathcal{R})}$ l'ensemble des $\mathcal{R}$-grammaires lexicalisées à arguments bornés par k. Pour tout $k' \geq 0$, la classe des langages de structures*

$$\{\ SL(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})} \text{ et } G \text{ est } k'\text{-valuée}\ \}$$

est apprenable pour tout $k' \geq 0$.

Démonstration. Simple application du corollaire 2.3. □

Rappelons la définition 4.14 : Une règle fortement contextuelle $R = A_1 \dots A_n \rightarrow B_1 \dots B_m$ est *strictement décroissante* si $n > m$.

Corollaire 5.3. *Soit k un entier positif, $\mathcal{R}$ un ensemble de règles strictement décroissantes à diminution limitée et $\mathcal{G}_{(k,\mathcal{R})}$ l'ensemble des $\mathcal{R}$-grammaires lexicalisées à arguments bornés par k. Pour tout $k' \geq 0$, la classe des langages de chaînes*

$$\{\ L(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})} \text{ et } G \text{ est } k'\text{-valuée}\ \}$$

a l'élasticité finie pour tout $k' \geq 0$.

Démonstration. Rappelons que $L(G) = \{\ prod(S) \mid S \in SL(G)\ \}$ (définition 4.8). Soit R la relation définie sur $\Sigma^* \times SL(\mathcal{R})_\Sigma$ par $x\,R\,S$ si et seulement si $x = prod(S)$ (autrement dit x est la phrase représentée par la structure S) : on remarque que pour toute $\mathcal{R}$-grammaire G $R^{-1}[SL(G)] = \{\, x \in \Sigma^* \mid \exists S \in SL(G)$ tel que $x = prod(S)\} = L(G)$. Soit $G \in \mathcal{G}_{(k,\mathcal{R})}$ une $\mathcal{R}$-grammaire k'-valuée. $\mathcal{R}$ est un ensemble de règles strictement décroissantes, donc pour tout x appartenant à $L(G)$ le nombre de $\mathcal{R}$-structures telles que $x = prod(S)$ est borné : en effet, si $|x| = n$, une telle structure contient n nœuds de règles locales et au maximum $n-1$ nœuds de règles universelles. La relation R est donc finiment valuée, or d'après la proposition 5.1 pour tout $k' \geq 0$ la classe $\{\ SL(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})}$ et G est k'-valuée $\}$ a l'élasticité finie. Par conséquent, d'après la proposition 2.14, la classe $\{\ L(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})}$ et G est k'-valuée $\}$ a aussi l'élasticité finie. □

Alternativement, ce corollaire peut être démontré en exhibant directement une borne sur la taille de chaque type de la grammaire réduite par rapport à D : soit $\langle t_1, \ldots, t_n\rangle = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n = \langle s\rangle$ une dérivation complète d'une chaîne $x \in D$. Comme chaque pas de dérivation est effectué avec une règle strictement décroissante, on a $|\alpha_i| > |\alpha_{i+1}|$ pour tout i, donc $n \leq |x| \leq M_D = max(\{\ |y| \mid y \in D\ \})$. En utilisant la proposition 5.10 qui donne $\|\alpha_i\| \leq \|\alpha_{i+1}\| + M_{(k,\mathcal{R})}$, on obtient $\|\alpha_i\| \leq (n-i) \times M_{(k,\mathcal{R})} + \|s\|$, soit pour $i = 0$

$$\sum_{i=1}^{n} \|t_i\| = \|\alpha_0\| \leq n \times M_{(k,\mathcal{R})} + \|s\| \leq M_D \times M_{(k,\mathcal{R})} + \|s\|.$$

Ainsi le nombre de grammaires de moins de k' règles réduites par rapport à D est borné, ce qui montre la densité finie bornée selon $\subseteq$ de $\langle \Sigma^*, \mathcal{G}_{(k,\mathcal{R})}, L\rangle$, avec pour conséquence (par le théorème 5.1) l'élasticité finie de la classe $\{\ L(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})}$ et G est k'-valuée $\}$.

Corollaire 5.4. *Soit k un entier positif, $\mathcal{R}$ un ensemble de règles strictement décroissantes à diminution limitée et $\mathcal{G}_{(k,\mathcal{R})}$ l'ensemble des $\mathcal{R}$-grammaires lexicalisées à arguments bornés par k. Pour tout $k' \geq 0$, la classe des langages de chaînes*

$$\{\ L(G) \mid G \in \mathcal{G}_{(k,\mathcal{R})} \textit{ et } G \textit{ est } k'\textit{-valuée}\ \}$$

est apprenable pour tout $k' \geq 0$.

Démonstration. Simple application du corollaire 2.3. □

5.3.3 Classes de $\mathcal{R}$-grammaires à arguments bornés apprenables

Nous proposons ci-dessous quelques exemples de classes de $\mathcal{R}$-grammaires auxquelles les résultats positifs d'apprenabilité sur les grammaires à arguments bornés démontrés plus haut sont appliquables. Certaines de ces applications reprennent des résultats d'apprenabilité déjà démontrés séparément pour un système grammatical particulier. Mais même dans ce cas le fait que ces résultats puissent être déduits directement à l'aide de notre méthode présente l'intérêt de souligner les similarités en termes d'apprenabilité des différentes classes concernées. En ce sens, nos résultats peuvent être vus comme une

généralisation de différentes techniques de démonstration de l'apprenabilité, mettant en avant les points communs qui caractérisent ces classes de langages.

Remarque : tous les résultats présentés ci-dessous pour les grammaires à arguments bornés par k (pour tout $k \geq 0$) incluent le cas des grammaires plates (c'est à dire les grammaires à arguments bornés par 1), présenté dans [78].

5.3.3.1 Grammaires catégorielles classiques et avec itérations

Nous avons vu dans l'exemple 5.2 que les règles des grammaires AB vérifient les conditions des règles à diminution limitée. On peut donc déduire directement du corollaire 5.2 que les langages de $\mathcal{R}$-structures des grammaires AB k-valuée à arguments bornés par k' sont apprenables, pour tous entiers positifs k et k'. Comme de plus les règles des grammaires AB sont strictement décroissantes, le corollaire 5.4 implique que les langages de chaînes des grammaires AB k-valuée à arguments bornés par k' sont apprenables, pour tous entiers positifs k et k'.

Ce résultat rejoint en partie celui obtenu par BESOMBES et MARION dans [18]. Dans cet article, les auteurs ont montré que la classe de langages des grammaires AB réversibles est apprenable. Le point commun entre leur résultat et le nôtre se situe dans la restriction aux grammaires plates (appelées grammaires compactes dans [18]), c'est-à-dire aux grammaires AB constituées uniquement de types dont les arguments sont bornés par 1. BESOMBES et MARION montrent que la sous-classe des grammaires plates réversibles est apprenable en totalité, tandis que nous ne ne montrons l'apprenabilité que pour des grammaires dont le nombre de règles est borné. À l'inverse, la classe dont nous prouvons l'apprenabilité n'est pas limitée aux grammaires à arguments bornés par 1, mais inclut toutes les grammaires dont les arguments sont bornés par un entier k. Dans le même article, BESOMBES et MARION démontrent que pour toute grammaire AB il existe une grammaire AB plate dont le langage est identique[7].

KANAZAWA avait démontré l'apprenabilité des grammaires AB k-valuées aussi bien à partir de structures que de chaînes (voir partie 3.3.2). Or il n'existe pas de constante k' telle que toutes les grammaires AB k-valuées soient des grammaires à arguments bornés par k', ce qui signifie que la classe des grammaires AB k-valuées n'est pas incluse dans celle des grammaires AB k-valuées à arguments bornés. Par conséquent le résultat que nous obtenons ici dans le cas des grammaires AB est plus faible que celui de KANAZAWA. Nous verrons dans le chapitre 6 un autre type de généralisation permettant au contraire d'inclure en totalité le résultat de KANAZAWA.

Nous avons proposé dans la partie 4.3.2 une extension simple des grammaires AB nommée *grammaires catégorielles avec itérations*. Nous en rappelons ci-dessous les règles universelles[8] :

[7] Cependant, cela n'implique ni la réversibilité ni l'existence d'une borne sur le nombre de règles d'une telle grammaire. On ne peut donc bien sûr rien en déduire quant à l'apprenabilité de la classe des langages de grammaires AB en général.

[8] Nous ne donnons ici que les règles pour les opérateurs $/, /^?, /^*, /^+$, les autres règles étant symétriques.

$$
\begin{array}{lll}
(R_{FA}) & A/B \quad B \rightarrow A & (\text{avec } Var(R_{FA}) = \{A, B\}) ; \\
(R_{BA}) & B \quad B\backslash A \rightarrow A & (\text{avec } Var(R_{BA}) = \{A, B\}) ; \\
(R_{/^*_{it}}) & A/^*B \quad B \rightarrow A/^*B & (\text{avec } Var(R_{/^*_{it}}) = \{A, B\}) ; \\
(R_{/^*_{\varepsilon}}) & A/^*B \rightarrow A & (\text{avec } Var(R_{/^*_{\varepsilon}}) = \{A, B\}) ; \\
(R_{/^+_{it}}) & A/^+B \quad B \rightarrow A/^+B & (\text{avec } Var(R_{/^+_{it}}) = \{A, B\}) ; \\
(R_{/^+_{1}}) & A/^+B \quad B \rightarrow A & (\text{avec } Var(R_{/^+_{1}}) = \{A, B\}) ; \\
(R_{/^?_{1}}) & A/^?B \quad B \rightarrow A & (\text{avec } Var(R_{/^?_{1}}) = \{A, B\}) ; \\
(R_{/^?_{\varepsilon}}) & A/^?B \rightarrow A & (\text{avec } Var(R_{/^?_{\varepsilon}}) = \{A, B\}) ;
\end{array}
$$

Les positions arguments des opérateurs sont définis de manière similaire à celle des opérateurs des grammaires AB, à savoir :

- $arg_{/^*}(1) = arg_{/^+}(1) = arg_{/^?}(1) = 0$ tandis que $arg_{/^*}(2) = arg_{/^+}(2) = arg_{/^?}(2) = 1$, et
- $arg_{\backslash^*}(1) = arg_{\backslash^+}(1) = arg_{\backslash^?}(1) = 0$ tandis que $arg_{\backslash^*}(2) = arg_{\backslash^+}(2) = arg_{\backslash^?}(2) = 1$.

On remarque que les conditions des règles à diminution limitée (définition 5.9) sont encore une fois respectées pour toutes ces règles. Ainsi, pour toute famille de $\mathcal{R}$-grammaires définie par cet ensemble de règles (ou un sous-ensemble de celui-ci), la classe des grammaires k-valuées à arguments bornés par k est apprenable à partir de $\mathcal{R}$-structures, pour tous $k, k' \geq 0$ (selon le corollaire 5.2). En revanche cette classe n'est pas nécessairement apprenable à partir de chaînes, parce que les règles $R_{/^*_{\varepsilon}}$ et $R_{/^?_{\varepsilon}}$ ne remplissent pas les conditions des règles strictement décroissantes : il faudrait pour cela qu'il y ait au moins deux types dans la partie gauche de la règle. En fait l'opérateur $/^+$ est le seul qu'il est possible d'ajouter aux langages des grammaires AB sans perte de l'apprenabilité (voir [15] pour plus de détails sur ce point).

5.3.3.2 Grammaires catégorielles combinatoires

COSTA FLORÊNCIO a proposé dans [32] une condition suffisante pour l'élasticité finie des grammaires combinatoires générales (sous la forme définie par KANAZAWA dans [60]). Le critère qu'il propose est basé sur la réduction de la classe de langages étudiée à celle des langages de grammaires AB k-valuées, pour laquelle l'élasticité finie est déjà démontrée par KANAZAWA (voir partie 3.3.2) : il montre l'existence d'une relation finiment valuée entre les deux classes, et applique ensuite le théorème de KANAZAWA (proposition 2.14) pour étendre l'élasticité finie de l'une à l'autre. Son critère diffère du nôtre par de nombreux aspects, car la méthode utilisée nécessite qu'il soit possible de transformer l'ensemble de règles universelles étudié en règles de forme très similaire à celle des règles des grammaires AB. Mais son critère a l'avantage de ne pas imposer de restriction sur la forme des types dans les grammaires, contrairement au nôtre (types à arguments bornés). De fait les deux méthodes mènent à deux résultats distincts, chacun permettant l'apprentissage de classes que l'autre ne permet pas.

COSTA FLORÊNCIO se place dans la continuité de la question laissée ouverte par KANAZAWA, concernant l'élasticité finie de certaines classes de grammaires combinatoires générales. Comme KANAZAWA envisageait cette question dans la perspective des grammaires catégorielles combinatoires de

STEEDMAN (voir partie 1.3.2.3), c'est avec ce formalisme que COSTA FLORÊNCIO illustre son résultat d'apprenabilité : il montre que certaines classes de langages définies par un sous-ensemble des règles universelles des GCG sont apprenables. Mais l'une des règles usuelles des CCG, la règle de composition, ne peut pas être prise en compte par son critère, alors que ceci est possible avec le nôtre :

$>B$ (Forward Composition)	$A/B\ B/C \rightarrow A/C$	$Var(>B) = \{A, B, C\}$
$<B$ (Backward Composition)	$C\backslash B\ B\backslash A \rightarrow C\backslash A$	$Var(<B) = \{A, B, C\}$
$>S$ (Forward Substitution)	$(A/B)/C\ B/C \rightarrow A/C$	$Var(>S) = \{A, B, C\}$
$<S$ (Backward Substitution)	$C\backslash B\ C\backslash(B\backslash A) \rightarrow C\backslash A$	$Var(<S) = \{A, B, C\}$

En effet, ces règles remplissent les conditions des règles à diminution limitée (définition 5.9). On remarque notamment que le type B dans la règle de composition $> B$ est en position argument dans la partie gauche, même s'il y a une autre occurence de B dans le type B/C (il en va de même pour la règle $< B$). Selon les corollaires 5.2 et 5.4, dans les familles de $\mathcal{R}$-grammaires utilisant sur ces règles, les classes de langages des grammaires k-valuées à arguments bornés par k' sont apprenables pour tout $k, k' \geq 0$. Cependant cet avantage ne doit pas masquer le fait que notre résultat d'apprenabilité est limité aux grammaires à arguments bornés, ce qui n'est pas le cas de celui de COSTA FLORÊNCIO.

Afin de compléter l'étude du cas des grammaires catégorielles combinatoires, il nous faut parler des règles de "montée des types" (*type raising*) :

$>T$:	$A \rightarrow B/(A\backslash B)$	$Var(>T) = \{A, B\}$
$<T$:	$A \rightarrow (B/A)\backslash B$	$Var(<T) = \{A, B\}$

Ces règles ne remplissent pas les conditions qui les rendraient éligibles à l'apprenabilité, ni dans le cadre proposé par COSTA FLORÊNCIO ni dans le nôtre (ces règles ne répondent pas à notre critère car B n'apparaît pas en partie gauche). Toutefois les types qui sont utilisables avec ces règles sont limités à un ensemble fini de catégories, de sorte qu'il est possible de "simuler" ces règles directement dans le lexique [103]. Cette solution permet que de telles classes de langages aient aussi l'élasticité finie, et soient donc apprenables [32].

En fait le résultat de COSTA FLORÊNCIO présente également un autre inconvénient par rapport au nôtre, qui sera discuté plus en détail dans la partie 6.6.

5.3.3.3 Grammaires catégorielles de liens

BÉCHET a démontré dans [14] que les grammaires de liens k-valuées sont apprenables à partir de chaînes, en utilisant également la propriété de densité finie bornée de SHINOHARA. Nous traitons plus précisément ici des grammaires catégorielles de liens (définies dans la partie 4.3.4), dont nous avons montré qu'elles sont un système de grammaires équivalent aux grammaires de liens[9] dans le cadre des GCG. Les deux règles des GCL sont :

[9] Cette équivalence étant limitée aux grammaires de liens sans cycles (voir partie 4.3.4.2).

$$R_l : \quad d(L, cons(c, R))\ \ d(cons(c, nil), nil) \rightarrow d(L, R) \quad Var(R_l) = \{c, L, R\}$$
$$R_r : \quad d(nil, cons(c, nil))\ \ d(cons(c, L), R) \rightarrow d(L, R) \quad Var(R_r) = \{c, L, R\}$$

L'ensemble des opérateurs étant $\mathcal{O}_{GCL} = \{\ \mathtt{d}/2,\ \mathtt{cons}/2,\ \mathtt{nil}/0\}$, on définit les positions arguments par $arg_{\mathtt{d}}(1) = arg_{\mathtt{d}}(2) = 0$, $arg_{\mathtt{cons}}(1) = 1$ et $arg_{\mathtt{cons}}(2) = 0$. Les règles R_l et R_r satisfont aux conditions des règles à diminution limitée (définition 5.9), car L et R, les seules variables qui ne sont pas en position argument dans la partie gauche, n'apparaissent pas dans la partie droite. Il est ainsi possible d'appliquer les corollaires 5.2 et 5.4 pour conclure à l'apprenabilité des grammaires catégorielles de liens k-valuées à arguments bornés, à partir de $\mathcal{R}$-structures comme à partir de chaînes. Or la définition d'origine des grammaires de liens (voir partie 1.3.3.4) n'autorise que des types primitifs dans les listes de connecteurs, et on peut constater dans le cas des GCL que les règles ne permettent pas non plus d'utiliser de types non atomiques[10] : on obtient donc ici le même résultat que BÉCHET[11].

En termes de langages, ce résultat est assez limité. Néanmoins, comme le formalisme des grammaires de liens est relativement éloigné de celui des grammaires catégorielles, il permet d'inclure le résultat obtenu par BÉCHET dans le cadre plus général des grammaires combinatoires générales. Ceci souligne les similitudes entre différentes classes de langages apprenables.

5.3.3.4 Grammaires catégorielles de dépendances

Les grammaires catégorielles de dépendances ont été présentées dans la partie 1.3.3.3 dans leur forme d'origine puis dans la partie 4.3.3 sous une forme simplifiée dans le modèle des grammaires combinatoires générales. Par rapport aux exemples précédents, celles-ci ont la particularité de n'être pas algébriques. Les opérateurs des GCD étant l'ensemble

$$\mathcal{O}_{GCD} = \{\ ^1(1),\ ^*(1),\ ^?(1),\ ^+(1),\ \#(1),\ /(2),\ \backslash(2),\ /\!\!/(2),\ \backslash\!\!\backslash(2),\ \swarrow(1),\ \searrow(1),\ \nwarrow(1),\ \nearrow(1)\ \},$$

les positions arguments de ces opérateurs sont définis de la manière suivante :

- $arg_{/}(1) = arg_{\backslash}(1) = arg_{/}(2) = arg_{\backslash}(2) = 0$;
- $arg_{/\!/}(1) = arg_{\backslash\!\backslash}(1) = arg_{/\!/}(2) = arg_{\backslash\!\backslash}(2) = 0$;
- $arg_{\#}(1) = 0$;
- $arg_{^1}(1) = arg_{^*}(1) = arg_{^?}(1) = arg_{^+}(1) = arg_{\swarrow}(1) = arg_{\searrow}(1) = arg_{\nwarrow}(1) = arg_{\nearrow}(1) = 1$.

Notons que, contrairement aux cas précédents, aucune position argument n'est définie pour les opérateurs de type $/$ et $\backslash$. Ceci est dû au fait que les arguments (au sens classique) de ces opérateurs seront toujours par définition de la forme C^1, C^*, $C^?$, C^+, $\swarrow C$, $\searrow C$, $\nwarrow C$ ou $\nearrow C$. Ce sont donc ces opérateurs qui sont définis avec une unique position argument, ce qui permet qu'un type de la forme $((A/B^1)/C^*)/D^+$ appartienne bien à l'ensemble $ArgB(\mathcal{V}, 1)$, autrement il s'agit d'un type plat.

Nous rappelons ci-dessous les règles universelles des GCD :

[10] Au sens où toute dérivation faisable avec des types non primitifs dans les liste de connecteurs le serait aussi en n'utilisant que des types primitifs.

[11] À la restriction due à l'équivance près.

$$\mathcal{R}_{GCD} = \left\{ \begin{array}{llllll} (R_L) & C & C^1\backslash\alpha & \rightarrow & \alpha & (\text{avec } Var(R_L) = \{C, \alpha\}) ; \\ (R_I) & C & C^*\backslash\alpha & \rightarrow C^*\backslash\alpha & & (\text{avec } Var(R_I) = \{C, \alpha\}) ; \\ (R_R) & C & C^+\backslash\alpha & \rightarrow C^*\backslash\alpha & & (\text{avec } Var(R_R) = \{C, \alpha\}) ; \\ (R_O) & C & C^?\backslash\alpha & \rightarrow \alpha & & (\text{avec } Var(R_O) = \{C, \alpha\}) ; \\ (R_{\Omega_*}) & C^*\backslash\alpha & \rightarrow \alpha & & & (\text{avec } Var(R_{\Omega_*}) = \{C, \alpha\}) ; \\ (R_{\Omega_?}) & C^?\backslash\alpha & \rightarrow \alpha & & & (\text{avec } Var(R_{\Omega_?}) = \{C, \alpha\}) ; \\ (R_{A_1}) & C & (\swarrow C)\backslash\!\backslash\alpha & \rightarrow & \#(\swarrow C) \quad \alpha & (\text{avec } Var(R_{A_1}) = \{C, \alpha\}) ; \\ (R_{A_2}) & C & (\searrow C)\backslash\!\backslash\alpha & \rightarrow & \#(\searrow C) \quad \alpha & (\text{avec } Var(R_{A_2}) = \{C, \alpha\}) ; \\ (R_{D_1}) & \#(\swarrow C) & (\nwarrow C)\backslash\alpha & \rightarrow & \alpha & (\text{avec } Var(R_{D_1}) = \{C, \alpha\}) ; \\ (R_{D_2}) & \#(\swarrow C) & \alpha & \rightarrow & \alpha \quad \#(\swarrow C) & (\text{avec } Var(R_{D_2}) = \{C, \alpha\}) \end{array} \right\}$$

Compte tenu des définitions des positions arguments, la catégorie C est bien en position argument (dans la partie gauche) dans toutes ces règles. Comme ces règles sont contextuelles et que tous les types qui y apparaissent sont plats, elles vérifient les critères des règles à diminution limitée (définition 5.9)). On peut donc déduire du corollaire 5.2 que les grammaires catégorielles de dépendances k-valuées à arguments bornés sont apprenables à partir de $\mathcal{R}$-structures. Comme dans le cas des grammaires de liens, les catégories utilisées comme arguments dans les GCD sont par définition toutes atomiques, donc la limitation de notre résultat aux grammaires à arguments bornées n'a aucune conséquence. Cependant, on ne peut rien dire ici de l'apprenabilité des langages de chaînes, puisque les règles R_{Ω_*}, $R_{\Omega_?}$, R_{A_1}, R_{A_2} et R_{D_2} ne sont pas strictement décroissantes.

Dans [15], il est démontré que les grammaires catégorielles de dépendances k-valuées sont apprenables à partir de structures, mais aussi qu'elles sont apprenables à partir de chaînes en supprimant seulement les règles concernant les opérateurs $C^?$ et C^*. En revanche, il y est aussi démontré que les GCD ne sont pas apprenables à partir de chaînes si l'on inclut ces deux opérateurs (et leurs règles). Ainsi notre résultat rejoint ceux démontrés dans [15], mais en partie seulement.

5.4 Conclusion

Dans [60], KANAZAWA montrait que pour toute grammaire AB k-valuée il existe une grammaire algébrique équivalente dont le nombre de règles est également borné. Il est possible qu'une équivalence de même type existe entre une grammaire appartenant à une classe de $\mathcal{R}$-grammaires k-valuée à arguments bornés et une grammaire contextuelle dont le nombre de règles est borné. Dans cette hypothèse, l'apprenabilité des langages de chaînes des $\mathcal{R}$-grammaires k-valuées à arguments bornés est incluse dans le résultat déjà obtenu par SHINOHARA sur les grammaires contextuelles dont le nombre de règles est limité (voir partie 2.5). Toutefois, non seulement notre résultat porte aussi sur les langages de structures, mais surtout il clarifie comment l'apprenabilité des langages contextuels peut s'appliquer à différents formalismes plus complexes. Il s'agit d'un point important pour les langues naturelles en particulier, puisque l'aspect structurel des phrases est primordial pour ces dernières.

Nous avons montré dans ce chapitre que les critères des règles à diminution limitée sont une condition suffisante pour l'apprenabilité de toute classe de $\mathcal{R}$-grammaires à arguments bornés. Ces critères imposés sur les règles universelles sont relativement souples, mais la limitation aux grammaires à arguments bornés est en fait un critère technique plus contraignant. En effet, cette condition posée a priori sur les classes de grammaires est un inconvénient assez sérieux pour certaines classes de $\mathcal{R}$-grammaires. Même si elle permet de prendre en compte des classes assez vastes de grammaires (et des formalismes relativement variés), cette contrainte n'a pas de sens en termes linguistiques et semble par ailleurs difficilement utilisable en pratique, puisqu'il est nécessaire de définir par avance une borne difficilement prédictible. Au niveau de la stricte apprenabilité, cela ne constitue pas un problème très grave, car on peut vraisemblablement prendre une borne suffisamment élevée qui convienne à toutes les langues naturelles. Mais il est clair également que la complexité de tout algorithme d'apprentissage d'une telle classe de langages dépendrait de cette borne. Dans le chapitre 6, nous proposons un autre résultat d'apprenabilité sur les grammaires combinatoires générales, qui n'est pas soumis à ce type de contrainte.

CHAPITRE 6

Apprenabilité des GCG par consommation d'arguments

NOUS AVONS DONNÉ DANS LE CHAPITRE PRÉCÉDENT UN PREMIER CRITÈRE, PORTANT SUR LES RÈGLES UNIVERSELLES, QUI GARANTIT L'ÉLASTICITÉ FINIE DES CLASSES DE GRAMMAIRES COMBINATOIRES GÉNÉRALES LE VÉRIFIANT. MAIS CE CRITÈRE NE PERMET PAS D'ENGLOBER TOTALEMENT LE RÉSULTAT PRINCIPAL DE KANAZAWA SUR LES GRAMMAIRES AB, CAR L'APPROCHE UTILISÉE NÉCESSITE UNE BORNE SUR LA TAILLE DE CERTAINS TYPES. DANS CE CHAPITRE, NOUS PROPOSONS UN NOUVEAU CRITÈRE QUI RÉPOND À CETTE PARTIE DU PROBLÈME. LE PRINCIPE CONSISTE CETTE FOIS-CI À CONSERVER AUTANT QUE POSSIBLE LES SPÉCIFICITÉS DES GRAMMAIRES AB DONT L'ÉLASTICITÉ FINIE DÉPEND, DE MANIÈRE À SUIVRE LA MÉTHODE EMPLOYÉE PAR KANAZAWA. C'EST POURQUOI LE CRITÈRE SERA BEAUCOUP PLUS STRICT SUR LA FORME DES RÈGLES, CE QUI LIMITE LA VARIÉTÉ DES FORMALISMES CONCERNÉS. EN CONTREPARTIE, NOUS OBTENONS SUR LES GRAMMAIRES VÉRFIANT CE CRITÈRE UN RÉSULTAT DE MÊME TYPE QUE CELUI DE KANAZAWA, INDÉPENDANT DE TOUTE CONTRAINTE SPÉCIFIQUE AU PROBLÈME DE L'APPRENABILITÉ. NOUS VERRONS AUSSI QUE LA FAIBLE ÉTENDUE DE CE RÉSULTAT EST LIÉE AUX LIMITES MÊMES DE L'APPROCHE UTILISÉE, EN EXHIBANT UN CONTRE-EXEMPLE POUR UN TYPE DE RÈGLES DONT LA FORME EST TRÈS PROCHE DE CELLE DES GRAMMAIRES AB.

6.1 Introduction

On a vu l'intérêt du résultat de KANAZAWA concernant l'apprenabilité des grammaires AB k-valuées à partir de simples phrases, et le rôle important de l'élasticité finie des langages de FA-structures des grammaires AB rigides dans ce résultat. En poursuivant la démarche de KANAZAWA [60], nous avons aussi montré dans la partie 4.4 que tous les langages de $\mathcal{R}$-structures de $\mathcal{R}$-grammaires rigides sont apprenables. Mais il reste à savoir quelles sous-classes de ces langages ont l'élasticité finie, de façon à étendre l'apprenabilité aux langages k-valués et (sous certaines conditions) aux langages de chaînes (c'est-à-dire de simples phrases).

Nous continuons donc dans ce chapitre l'étude de la question laissée ouverte par KANAZAWA, qui

consiste à caractériser les sous-classes de grammaires combinatoires générales ayant l'élasticité finie. Comme nous l'avons expliqué dans le chapitre 5, cette question présente un grand intérêt pour l'apprenabilité des langues naturelles, qui nécessite à la fois l'usage de formalismes grammaticaux appropriés, une expressivité assez large et la possibilité d'apprendre à partir de phrases brutes. Les GCG offrent un cadre uniforme qui permet de traiter cette question à l'aide de critère(s) sur les règles universelles, cette généricité autorisant une réutilisabilité relativement importante des résultats obtenus. On peut assez facilement montrer que certaines classes de $\mathcal{R}$-grammaires n'ont pas l'élasticité finie, comme en témoigne l'exemple 3.2. En revanche la découverte de conditions suffisantes pour l'élasticité finie de classes de grammaires complexes n'est pas triviale. Naturellement, on cherche d'abord à extraire des cas d'élasticité finie déjà démontrés les caractéristiques qui sont essentielles pour cette élasticité, de façon à s'en servir comme base pour des classes de grammaires plus vastes. Ainsi nous avons repris dans le chapitre précédent l'approche que SHINOHARA [95] a utilisée pour les grammaires contextuelles k-bornées. Dans le même ordre d'idées, nous proposons maintenant de suivre l'approche utilisée par KANAZAWA pour les grammaires AB. Nous verrons que celle-ci n'est pas sans rapport avec celle de SHINOHARA, puisqu'on peut avantageusement l'inclure dans le concept de densité finie bornée généralisée[1]. Pourtant cette approche présente une différence notable par rapport à celle développée dans le chapitre 5 : au lieu de restreindre les $\mathcal{R}$-grammaires concernées par un critère ne conservant que les types à arguments bornés, on limite ici les grammaires à celles dont la forme des dérivations permet justement de ne pas avoir besoin de ce type de contrainte. Il en résulte une apprenabilité plus étendue et plus simple pour les classes de grammaires respectant ce critère : l'exemple le plus évident est celui des grammaires AB k-valuées, qui respectent le critère et sont par conséquent apprenables en totalité (c'est-à-dire pas seulement celles à arguments bornés)[2].

Mais s'abstraire d'une telle contrainte tout en restant dans le cadre de l'élasticité finie oblige à limiter de façon beaucoup plus stricte la variété des formes de grammaires apprenables. Nous verrons que la forme des règles doit être vraiment très proche de celle des grammaires AB pour que le résultat s'y applique, ce qui élimine la grande majorité des formalismes grammaticaux représentables en GCG. Bien entendu les résultats que nous proposons ne sont que des conditions suffisantes de l'élasticité finie, ce qui signifie qu'il est possible que des classes beaucoup plus vastes aient elles aussi l'élasticité finie. Néanmoins, si l'on part du principe que le résultat de KANAZAWA est jusqu'à présent l'un des plus intéressants sous ces conditions, donc qu'apparemment c'est en élargissant les recherches autour de son résultat qu'on peut trouver des classes ayant le même genre de propriétés, alors on découvre assez vite les limites de cette approche : le contre-exemple 6.3 (p. 188) montre une classe de grammaires très proche des grammaires AB qui n'a pas l'élasticité finie. On est donc en droit de supposer que la frontière de l'élasticité finie (sans contrainte de borne sur les types) n'est pas très éloignée du résultat positif que nous obtenons. Ceci tend maheureusement à montrer que le cas positif des grammaires AB pour l'élasticité finie est plutôt une exception parmi les GCG.

[1] Ce qui illustre aussi le fait que le concept de densité finie bornée selon une relation autre que l'inclusion simple (présenté dans la partie 5.2) a un sens.

[2] En effet, on retrouve très logiquement le résultat de KANAZAWA dans la généralisation de sa méthode.

Remarque : Dans ce chapitre nous étudions uniquement le cas des $\mathcal{R}$-grammaires algébriques lexicalisées. Dans la mesure où les contraintes sur les grammaires étudiées dans ce chapitre sont nombreuses, nous ne le préciserons pas à chaque étape afin de ne pas surcharger les propositions par la répétition de cette information. En revanche nous détaillons les démonstrations de la plupart des propositions nécessaires au résultat final, de manière à permettre au lecteur de visualiser plus facilement les points clé qui mènent à l'élasticité finie. Cela est assez important pour une démonstration d'apprenabilité, car on peut en tirer l'idée générale de l'algorithme d'apprentissage correspondant (même s'il s'agirait en l'occurrence d'un algorithme absolument inutilisable en pratique).

6.2 Propriétés des $\mathcal{R}$-grammaires avec la relation $\sqsubseteq$

6.2.1 Types utiles et $\mathcal{R}$-grammaires faiblement réduites

Dans [60], KANAZAWA a recours à la notion de *types utiles* pour associer une seule grammaire à un ensemble (généralement infini) de grammaires dont les langages sont identiques. Il s'agit d'une sorte de forme normale basée sur l'existence de types dits *inutiles* dans les grammmaires. On peut démontrer que de tels types peuvent être éliminés sans changer le langage de la grammaire (d'où leur nom). De même que dans les grammaires AB étudiées par KANAZAWA, on définit ci-dessous la notion de type utile/inutile dans les $\mathcal{R}$-grammaires en général, et on démontre les mêmes équivalences de langages.

Globalement le principe demeure identique au cas des grammaires AB : un type "inutilement complexe" par rapport à la structure du langage doit être considéré comme inutile, de manière à se ramener à la forme la plus simple de grammaire "sans types inutiles". Cependant la définition de "l'utilité" des types était simplifiée dans le cas des grammaires AB par les spécificités des structures de ces grammaires, là où la généralisation aux $\mathcal{R}$-grammaires doit capturer les propriétés essentielles de cette notion sans restriction à une forme de structures (ou de règles) particulière.

6.2.1.1 Définitions

Dans la suite on notera $t[a \mapsto b]$ le type résultant du remplacement dans t de toutes les occurrences du type a par le type b. Il ne s'agit pas d'une substitution au sens classique (voir définition 1.6) dans la mesure où ce remplacement ne garantit pas que

$$(f(t_1, t_2, \ldots, t_n))[a \mapsto b] = f(t_1[a \mapsto b], t_2[a \mapsto b], \ldots, t_n[a \mapsto b]),$$

car a peut être un type complexe.

Définition 6.1 (Remplacement de type)**.** Le remplacement de a par b dans t, noté $t[a \mapsto b]$, est défini de la façon suivante :

- si $t = a$, alors $t[a \mapsto b] = b$.
- si $t \neq a$, alors
  - si t est primitif, $t[a \mapsto b] = t$.
  - sinon $t = f(t_1, t_2, \ldots, t_n)$: $t[a \mapsto b] = f(t_1[a \mapsto b], t_2[a \mapsto b], \ldots, t_n[a \mapsto b])$.

Définition 6.2 (Type utile)**.** Soit G une $\mathcal{R}$-grammaire algébrique lexicalisée dans le système $\langle \mathcal{O}, \mathcal{R}, s \rangle$. Un type $t \in Tp$ est *utile* pour G si t est un sous-type de s[3] ou s'il existe une règle $R \in \mathcal{R}$, $R = A_1 \dots A_n \rightarrow A_0$, et une substitution $\sigma : Var(R) \mapsto Tp$ tels que

- $\sigma(A_i) \in FT(G)$ pour tout $i \geq 0$
- il existe j ($0 \leq j \leq n$) et un sous-type u dans A_j tels que
  - $\sigma(u) = t$,
  - et $u \in Var(R)$ si et seulement si t est primitif.[4]

On peut grossièrement résumer cette définition de la façon suivante : un type t n'est utile que s'il apparaît dans une dérivation de la grammaire à une position où il "joue un rôle". En effet l'existence d'un sous-type u dans A_j tel que $\sigma(u) = t$ garantit que t s'unifie au moins avec une partie du type A_j.

Définition 6.3 (Grammaire sans type inutile)**.** Un type t est *inutile* pour une grammaire G si t n'est pas utile pour G.

Une grammaire G est dite *sans type inutile* si tout type $t \in STLex(G)$ (définition 4.17, p. 99) est utile pour G. À l'inverse, G *contient un type inutile* s'il existe un type $t \in STLex(G)$ inutile.

Le fait qu'une grammaire ne contienne pas de type inutile interdit notamment la multitude de grammaires équivalentes que l'on peut créer à partir d'une grammaire en substituant n'importe quel type complexe (constitué de nouveaux types primitifs) à un type primitif. On notera également que si un type t est inutile pour une grammaire alors tout sous-type de t est également inutile pour celle-ci, par définition.

6.2.1.2 Propriété remarquable des grammaires $\sqsubseteq$-réduites

Lemme 6.1. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire, et soit t un type primitif tel qu'il existe une règle $w \triangleright t''$ dans G avec t un sous-type strict de t''.*

Si t est inutile pour G alors il existe un type $t' \in STLex(G)$ inutile pour G tel que t est un sous-type strict de t'.

Démonstration. t est un sous-type strict de t'', par conséquent il existe un sous-type t' dans t'' de la forme $t' = f(t_1, \dots, t_n)$, avec $t_k = t$, $1 \leq k \leq n$. On montre par contraposée que si t est inutile pour la grammaire G alors t' l'est aussi.

On suppose que t' est utile pour G : il existe $R \in \mathcal{R}$, $R = A_1 \dots A_n \rightarrow A_0$, et une substitution $\sigma : Var(R) \mapsto Tp$ tels que

- $\sigma(A_i) \in FT(G)$ pour tout $i \geq 0$
- il existe j ($0 \leq j \leq n$) et un sous-type u dans A_j tels que $\sigma(u) = t'$, et $u \in Var(R)$ si et seulement si t' est primitif.

Or t' n'est pas primitif par définition, donc $u \notin Var(R)$, autrement dit u est un type complexe. Comme $\sigma(u) = t' = f(t_1, \dots, t_n)$, u est nécessairement de la forme $u = f(u_1, \dots, u_n)$ avec $\sigma(u_i) = t_i$ pour

[3] Remarquons que dans ce cas t ne contient aucun type primitif de Pr.

[4] Remarque : cette condition est équivalente à "u primitif implique t primitif" car il est impossible que t soit primitif sans que u ne le soit, puisque $\sigma(u) = t$.

tout i. Ainsi il existe un sous-type u_k dans A_j tel que $\sigma(u_k) = t$, avec u_k primitif et t primitif : t est donc utile pour la grammaire G. □

Proposition 6.1. *Soit $\langle \mathcal{O}, \mathcal{R}, s\rangle$ un système de $\mathcal{R}$-grammaires algébriques lexicalisées et $\mathcal{G}_\mathcal{R}$ l'ensemble des $\mathcal{R}$-grammaires dans ce système.*

Si $G \in \mathcal{G}_\mathcal{R}$ est faiblement $\sqsubseteq$-réduite dans le système de grammaires $\langle SL(\mathcal{R}), \mathcal{G}_\mathcal{R}, SL\rangle$, alors G ne contient pas de type inutile.

Démonstration. On montre la contraposée. Supposons que $G = \langle \Sigma, Pr, \triangleright\rangle$ contienne un type inutile, c'est-à-dire qu'il existe un type $t' \in Lex(G)$ et un sous-type t dans t' tels que t n'est pas utile pour G. On montre dans les trois cas ci-dessous l'existence d'une grammaire $G' \sqsubset G$ telle que $SL(G) \subseteq SL(G')$.

Cas 1. t est un type primitif et $t' = t$, autrement dit il existe une règle $w \triangleright t$ dans G. Soit $G' = \langle \Sigma', Pr', \triangleright'\rangle$ la grammaire définie par $G' = G \setminus \{w \triangleright t\}$ (G' contient toutes les règles de G sauf la règle $w \triangleright t$). Clairement $G' \subset G$, ce qui implique par définition que $G' \sqsubset G$. On montre qu'il n'existe aucune structure de dérivation P pour G utilisant la règle $w \triangleright t$:

D'abord, $t \neq s$ puisque t est inutile, donc la structure $[](s, w)$ (qui est le seul cas possible de structure de dérivation correcte sans règle universelle) n'est pas correcte pour G. On montre ensuite que s'il existe une structure de G utilisant t en entrée d'une règle universelle alors t n'est pas inutile : supposons qu'il existe une règle $R = A_1 \dots A_n \to A_0$ et une substitution σ telle que $\sigma(A_j) = t \in FT(G)$. Dans ce cas il existe un sous-type u dans A_j, défini par $u = A_j$, tel que $\sigma(u) = t$, ce qui implique que u est primitif ($u \in Var(R)$) puisque t est primitif. Ces conditions montrent que t est utile pour G, ce qui contredit l'hypothèse de départ. Par conséquent la règle $w \triangleright t$ n'est jamais utilisée dans G, donc toute structure de G est réalisable dans G' : $SL(G) \subseteq SL(G')$.

Cas 2. t est un type primitif et $t' \neq t$, autrement dit t est un sous-type strict de t'. D'après le lemme 6.1, il existe un type $t'' \in STLex(G)$ inutile pour G, tel que t est un sous-type strict de t''. Ainsi t'' est un type complexe inutile pour G : on démontre dans le cas 3 ci-dessous que cela implique l'existence d'une grammaire $G' \sqsubset G$ telle que $SL(G) \subseteq SL(G')$.

Cas 3. t n'est pas un type primitif. Soit x une nouvelle variable, et $Pr' = Pr \cup \{x\}$. Soit $G' = \langle \Sigma', Pr', \triangleright'\rangle$ la grammaire définie par $\Sigma' = \Sigma$ et $w \triangleright' u[t \mapsto x]$ si et seulement si $w \triangleright u$ (avec $u[t \mapsto x]$ le type u dans lequel toute occurrence de t est remplacée par x). D'après cette définition, il existe une substitution $\tau : Pr' \mapsto Pr$, définie par $\tau(y) = y$ pour tout $y \in Pr$ et $\tau(x) = t$, telle que $\tau[G'] = G$. La substitution τ est fidèle, car $\tau(y_1) \neq \tau(y_2)$ pour tous $y_1 \neq y_2 \in Pr$, $\tau(x) = t \neq y$ pour tout $y \in Pr$, et pour tout $u \in Lex(G')$ il n'existe aucune occurrence de t dans u par définition : ainsi pour tout couple $u, u' \in Lex(G')$ on a $\tau(u) \neq \tau(u')$. Ainsi $\tau[G'] = G$ et τ est fidèle, donc $G' \sqsubset G$. On montre maintenant que $SL(G) \subseteq SL(G')$:

Soit T une $\mathcal{R}$-structure et P une instance de T pour G. Soit $P' = P[t \mapsto x]$, c'est-à-dire la structure de dérivation définie comme la structure P dans laquelle à chaque nœud le type de ce nœud u est remplacé par $u[t \mapsto x]$. On montre par induction sur la hauteur h de P que la structure de dérivation P' est une instance de T pour G'.

- $h = 1$. $P = [](u, w)$, avec $w \triangleright u$ une règle de G. Par définition de G' il existe une règle $w \triangleright u[t \mapsto x]$, donc $P' = [\,](u[t \mapsto x], w)$ est une instance de T pour G.
- $h > 1$. $P = [R](t_0, P_1, \ldots, P_n)$, avec $R \in \mathcal{R}$, donc $P' = [R](t_0[t \mapsto x], P_1[t \mapsto x], \ldots, P_n[t \mapsto x])$. On suppose que la propriété est vraie pour toute structure de hauteur $h' < h$, donc pour tout i, $1 \leq i \leq n$, $P_i[t \mapsto x]$ est une structure correcte pour G'. Soit $t_i = racine(P_i)$ pour tout $i > 0$ et $t'_i = t_i[t \mapsto x]$ pour tout $i \geq 0$ (on a donc $t'_i = racine(P_i[t \mapsto x])$ pour tout $i > 0$). On montre que si $t_1 \ldots t_n \rightarrow_R t_0$, alors $t'_1 \ldots t'_n \rightarrow_R t'_0$. Soit $R = A_1 \ldots A_n \rightarrow A_0$. On suppose que $t_1 \ldots t_n \rightarrow_R t_0$, donc il existe une substitution $\sigma : Var(R) \mapsto Tp$ telle que $\sigma(A_i) = t_i \in FT(G)$ pour tout $i \geq 0$. Soit σ' la substitution définie sur $Var(R)$ par $\sigma'(v) = \sigma(v)[t \mapsto x]$ pour tout $v \in Var(R)$. On montre que pour tout $i \geq 0$ $\sigma'(A_i) = t'_i$:
  - si t n'est pas un sous-type de t_i, alors $\sigma'(A_i) = \sigma(A_i) = t_i = t'_i$.
  - sinon, supposons que $\sigma'(A_i) \neq t'_i$, c'est-à-dire $\sigma'(A_i) \neq (\sigma(A_i))[t \mapsto x]$. Dans $\sigma(A_i)$, comme toutes les occurrences de t qui apparaissent à l'intérieur d'une occurrence d'un $\sigma(v)$ ($v \in Var(R)$) sont remplacées par x, il doit exister au moins une occurrence de t dans $\sigma(A_i)$ qui n'est incluse dans aucune occurrence de $\sigma(v)$, quel que soit $v \in Var(R)$. Cela signifie que A_i contient un sous-type a tel que $\sigma(a) = t$ et $a \notin Var(R)$. Or t est un type complexe inutile, donc cela contredit la définition 6.2 : en effet tout sous-type a dans A_i tel que $\sigma(a) = t$ appartient nécessairement à $Var(R)$, puisque t est complexe. Cette contradiction invalide l'hypothèse selon laquelle $\sigma'(A_i) \neq t'_i$.

  Pour tout i, $0 \leq i \leq n$, $\sigma'(A_i) = t'_i = t_i[t \mapsto x]$, donc P' est bien une instance de T pour G.

Ainsi quelle que soit la hauteur de la structure P il existe une structure équivalente P' qui est une instance de T pour G'. En particulier si $T \in SL(G)$ il existe une structure de dérivation P' pour G' qui une instance de T. $racine(P') = racine(P)[t \mapsto x] = s[t'' \mapsto x] = s$ (car si t est inutile alors t n'est pas un sous-type de s). Par conséquent $T \in SL(G')$, et donc en général $SL(G) \subseteq SL(G')$.

Dans les trois cas précédents on a montré qu'il existe $G' \sqsubset G$ telle que $SL(G) \subseteq SL(G')$. De plus d'après la proposition 4.13 $SL(G') \subseteq SL(G)$, donc $SL(G) = SL(G')$. Par conséquent G n'est pas faiblement $\sqsubseteq$-réduite. □

Remarque : La réciproque est fausse en général, comme le montre le contre-exemple suivant dans le formalisme des grammaires catégorielles de liens (cf. partie 4.3.4) : la grammaire $G = \{\, x : \texttt{d([a],[])}\ ;\ y : \texttt{d([],[a])}\ ;\ z : \texttt{a}\}$ ne contient pas de type inutile car `a` est utile, en effet la dérivation complète

$$x\ y\ \rightarrow_{CLG}\ \texttt{d([a],[])}\ \texttt{d([],[a])}\ \rightarrow_{CLG}\ \texttt{d([],[])}$$

utilise `a` (dans les conditions requises par la définition 6.2). Mais il n'est pas difficile de voir que la dernière règle qui associe le type `a` à z ne peut jamais être utilisée, ce qui implique l'existence d'une grammaire $G' \sqsubset G$, définie simplement par $G' = \{\, x : \texttt{d([a],[])}\ ;\ y : \texttt{d([],[a])}\}$.

6.2.2 Densité finie bornée selon la relation $\sqsubseteq$

Proposition 6.2. *Soit $\mathcal{R}$ un ensemble de règles universelles et $\mathcal{G}_\mathcal{R}$ l'ensemble des $\mathcal{R}$-grammaires. $\sqsubseteq$ (voir définition 4.25) est une relation de réduction (voir définition 5.1) pour le système de grammaires $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}_\mathcal{R}, SL\rangle$.*

Démonstration.

- La relation $\sqsubseteq$ est un ordre partiel, d'après la proposition 4.12.
- $\forall G, G' \in \mathcal{G}_\mathcal{R}$, $G \sqsubseteq G' \Rightarrow SL(G) \subseteq SL(G')$, d'après la proposition 4.13.
- D'après le corollaire 4.2, $\{\ G' \in \mathcal{G}_\mathcal{R} \mid G' \sqsubseteq G\ \}$ est fini pour toute grammaire $G \in \mathcal{G}_\mathcal{R}$. Par conséquent $\sqsubseteq$ est un ordre bien fondé.

□

Nous montrons dans cette partie que toute classe de $\mathcal{R}$-grammaires a la densité finie bornée selon la relation $\sqsubseteq$. Il s'agit d'une partie importante de la démonstration de l'élasticité finie, mais d'une partie seulement. En effet, cela n'a d'intérêt que pour les classes $\mathcal{G}$ telles que $MaxRed(\mathcal{G}, \sqsubseteq, k)$ couvre un ensemble "pertinent" de grammaires, car l'apprenabilité de la classe $MaxRed(\mathcal{G}, \sqsubseteq, k)$ en elle-même est un simple résultat technique dénué de sens[5]. C'est pourquoi il faudra ensuite montrer que la relation de réduction $\sqsubseteq$ a certaines propriétés remarquables avec certaines classes précises de grammaires (en particulier, si les langages décrits par $\mathcal{G}$ sont tous inclus dans ceux de $MaxRed(\mathcal{G}, \sqsubseteq, k)$).

Proposition 6.3. *Soit $\langle \mathcal{O}, \mathcal{R}, s\rangle$ un système de $\mathcal{R}$-grammaires, $\mathcal{G}$ une classe de grammaires dans ce système et $D \subseteq SL(\mathcal{R})$ un ensemble fini de $\mathcal{R}$-structures. L'ensemble*

$$\{\ G \in \mathcal{G} \mid G \textit{ est } \sqsubseteq \textit{-réduite par rapport à } D\ \}$$

est fini.

Démonstration. On utilise les algorithmes GF, SG et k-SG définis dans la partie 4.4. Soit k le nombre de règles de la grammaire $GF(D)$: nous allons montrer que

$$\{\ G \in \mathcal{G} \mid G \text{ est } \sqsubseteq \text{-réduite par rapport à } D\ \} \subseteq \text{k-SG}(D).$$

Soit $G \in \mathcal{G}$ une $\mathcal{R}$-grammaire $\sqsubseteq$-réduite par rapport à D. $D \subseteq SL(G)$, donc d'après la proposition 4.19 il existe une substitution σ telle que $\sigma[GF(D)] \subseteq G$.

Montrons que $\sigma[GF(D)] = G$. Supposons au contraire que $\sigma[GF(D)] \subset G$: Soit τ la substitution identité. Clairement $\tau[\sigma[GF(D)]] \subseteq G$, et τ est évidemment une substitution fidèle, donc $\sigma[GF(D)] \sqsubseteq G$ avec $\sigma[GF(D)] \neq G$. Or $D \subseteq SL(\sigma[GF(D)])$ (par la proposition 4.19), donc il existe une grammaire $\sigma[GF(D)]$ telle que $D \subseteq SL(\sigma[GF(D)])$ et $\sigma[GF(D)] \sqsubset G$, ce qui contredit le fait que G soit $\sqsubseteq$-réduite par rapport à D. Par conséquent $\sigma[GF(D)] = G$.

$\sigma[GF(D)] = G$ implique $|G| = |\sigma[GF(D)]| \leq |GF(D)| = k$, donc G est une grammaire k-bornée. Ainsi selon la proposition 4.23 il existe une grammaire $G' \in$ k-SG(D) telle que $G' \sqsubseteq G$. $D \subseteq G'$ (par construction et selon la proposition 4.19), or G est $\sqsubseteq$-réduite par rapport à D donc il est impossible que $G' \sqsubset G$. Par conséquent $G' = G$ (modulo un éventuel renommage), donc $G \in$ k-SG(D). □

Corollaire 6.1. *Soit $\langle \mathcal{O}, \mathcal{R}, s\rangle$ un système de $\mathcal{R}$-grammaires, $\mathcal{G}$ une classe de grammaires sur un vocabulaire Σ dans ce système et $D \subseteq SL(\mathcal{R})_\Sigma$ un ensemble fini de $\mathcal{R}$-structures sur Σ.*

Le système de grammaires $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}, SL\rangle$ a la densité finie bornée selon $\sqsubseteq$.

[5] Rappelons que cette classe est constituée de l'ensemble des grammaires telles que toute séquence $G_1 \sqsubset G_2 \sqsubset \cdots \sqsubset G_n$ vérifie $n \leq k$.

Démonstration. Pour tout ensemble fini $D \subseteq SL(\mathcal{R})_\Sigma$, l'ensemble

$$\{\ G \in \mathcal{G} \mid G \text{ est } \sqsubseteq\text{-réduite par rapport à } D\ \}$$

est fini (d'après la proposition 6.3). A fortiori, pour tout $k \geq 0$ le sous-ensemble $\{\ G \in \mathcal{G} \mid G \in MaxRed(\mathcal{G}, \sqsubseteq, k)$ et G est $\sqsubseteq$-réduite par rapport à $D\}$ est fini, par conséquent l'ensemble $\{SL(G) | G \in MaxRed(\mathcal{G}, \sqsubseteq, k)$ et G est $\sqsubseteq$-réduite par rapport à $D\ \}$ est fini. Ainsi $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}, SL\rangle$ a la densité finie bornée selon $\sqsubseteq$, par définition. □

Proposition 6.4. *Soit $\mathcal{G}_k$ une classe de $\mathcal{R}$-grammaires telle que $|G| \leq k$ pour toute grammaire $G \in \mathcal{G}_k$.*

Soit $\{G_1, G_2, \ldots, G_n\} \subseteq \mathcal{G}_k$ un ensemble de grammaires tel que pour tout i ($1 < i \leq n$) il existe une substitution σ_i fidèle telle que $\sigma_i(G_{i-1}) = G_i$, avec $G_{i-1} \not\equiv G_i$.

S'il existe une constante C telle que $C \geq n$ pour tout ensemble vérifiant de telles conditions, alors il existe une constante C' telle que

$$\mathcal{G}_k \subseteq MaxRed(\mathcal{G}_k, \sqsubseteq, C').$$

Démonstration. Soit $\langle G'_1, G'_2, \ldots, G'_{n'}\rangle$ une séquence de grammaires telle que $G'_1 \sqsubset G'_2 \sqsubset \cdots \sqsubset G'_{n'}$, avec $G'_i \in \mathcal{G}_k$ pour tout i. On définit la fonction f sur $\{1, \ldots, n'\}$ par[6]

- $f(1) = 1$;
- si $i > 1$, $f(i) = min(\{\ j \mid |G'_{f(i-1)}| < |G'_j|\ \})$.

Soit $m = max(\{\ i \in \{1 \ldots n'\} \mid f(i) \leq n'\ \})$ (ainsi $i \leq m$ si et seulement si $f(i) \neq n' + 1$). Par définition de f, pour tout i, $1 \leq i < m$, il existe une subsitution fidèle[7] σ telle que $\sigma[G'_{f(i)}] \subset G'_{f(i+1)}$, donc $|G'_{f(i)}| < |G'_{f(i+1)}|$. Comme $|G'_i| \leq k$ pour tout i (car $G'_i \in \mathcal{G}_k$), on obtient $m \leq k + 1$.

Pour tout i, $1 \leq i \leq m$, et tout j, $f(i) < j < f(i + 1)$, par définition de f on a $|G'_{f(i)}| = |G'_j|$, donc il existe une substitution σ fidèle telle que $\sigma[G'_{f(i)}] = G'_j$ et $G'_{f(i)} \not\equiv G'_j$. Soit $G_l = G'_{l+f(i)-1}$ pour tout l, $1 \leq l \leq f(i + 1) - f(i)$) : pour tout $l > 0$ il existe σ_l telle que $\sigma_l(G_{l-1}) = G_l$, et $\{G_1, \ldots, G_{f(i+1)-f(i)}\} \subseteq \mathcal{G}_k$, donc par hypothèse $f(i + 1) - f(i) \leq C$.

Soit $s_i = \langle G'_{f(i)}, G'_{f(i)+1}, \ldots, G'_{f(i+1)-1}\rangle$ pour tout i, $1 \leq i \leq m$: clairement la séquence $\langle G'_1, \ldots G'_{n'}\rangle$ est composée de la concaténation des sous-séquences $s_1 \bullet s_2 \bullet \cdots \bullet s_m$. Chacune de ces sous-séquences est de longueur inférieure ou égale à C et m est inférieur ou égal à $k + 1$, par conséquent $n' \leq C \times (k + 1)$. Soit $C' = (C + 1) \times k - 1$: $n' - 1 \leq C'$ pour toute grammaire $G' = G'_{n'} \in \mathcal{G}_k$ et tout ensemble $\{G'_1, \ldots, G'_{n'-1}\} \subseteq \mathcal{G}_k$ tel que $G'_i \sqsubset G'_{i+1}$ pour tout i, donc $\mathcal{G}_k \subseteq MaxRed(\mathcal{G}_k, \sqsubseteq, C')$. □

6.3 Grammaires par consommation d'arguments

Les grammaires par consommation d'arguments sont un sous-ensemble des $\mathcal{R}$-grammaires lexicalisées algébriques. Leur spécificité, traduite ci-dessous en termes de contraintes syntaxiques, est le fonctionnement par consommation d'arguments : les règles universelles n'autorisent qu'une seule forme de

[6] La fonction f est définie de telle sorte que $f(i)$ soit l'indice de la prochaine grammaire comportant strictement plus de règles que la grammaire d'indice $f(i - 1)$, ou $n + 1$ (indice hors bornes) s'il n'existe pas de telle grammaire.

[7] σ est nécessairement une substitution fidèle car $G'_{f(i)} \sqsubset G'_{f(i+1)}$.

dérivation, dans laquelle un type dit *consommateur* requiert un certain nombre d'arguments pour pouvoir se transformer en un sous-type. Les autres types présents en partie gauche d'une dérivation ne peuvent donc être que des arguments du type consommateur. Ces contraintes ont pour but de contrôler précisément la façon dont les types interagissent dans un ensemble de dérivation, afin d'en déduire certaines propriétés suffisantes pour l'élasticité finie. Bien entendu les grammaires AB sont des grammaires par consommation d'arguments, puisque l'on s'inspire de leurs propriétés pour les étendre à ces nouvelles classes de grammaires.

6.3.1 Opérateurs orientés

On considère ici des opérateurs munis (chacun) d'une fonction arg définissant les positions arguments de l'opérateur (voir définition 4.23).

Définition 6.4 (Opérateur orienté)**.** Un opérateur f d'arité n est dit *orienté* si
- Il existe h, $1 \leq h \leq n$, tel que $arg_f(h) = 0$,
- et $arg_f(i) = 1$ quel que soit $i \neq h$.

Soit $\mathcal{O}$ un ensemble d'opérateurs orientés et t un $\mathcal{O}$-terme sur Pr. $pr(t)$ est la fonction définie par
- Si $t \in Pr$, alors $pr(t) = t$
- Sinon, soit $t = f(t_1, \ldots, t_n)$ et soit h l'unique indice tel que $arg_f(h) = 0$: $pr(t) = t_h$.

$pr(t)$ est appelé le *type principal* de t.

Autrement dit la particularité d'un opérateur orienté est d'avoir une et une seule position principale, les autres positions éventuelles étant toutes des positions d'argument. Comme on le verra, il découle de cette définition de nombreuses propriétés. Afin de les étudier nous nous dotons des quelques outils définis ci-dessous, spécifiques aux opérateurs orientés.

Définition 6.5 (Tête et sous-types têtes)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs orientés et t un $\mathcal{O}$-terme sur Pr. Soit $head(t)$ la fonction définie par :
- $head(t) = t$ si $t \in Pr$
- sinon, $head(t) = head(pr(t))$.

$head(t)$ est appelée *la tête* du type t.

Soit $head^*(t)$ la fonction définie par :
- $head^*(t) = \{t\}$ si $t \in Pr$
- sinon, $head^*(t) = \{t\} \cup head^*(pr(t))$.

$head^*(t)$ est l'ensemble des *sous-type têtes* de t, autrement dit l'ensemble des sous-types obtenus en parcourant t en profondeur, et en ne considérant à chaque niveau que le type principal : cette série de types décrit la "branche principale" du type. On peut noter que de cette définition découlent les quelques propriétés immédiates ci-dessous :
- $head(t') = head(t)$ pour tout $t' \in head^*(t)$;
- $t = t'$ si et seulement si $head^*(t) = head^*(t')$;
- $t' \in head^*(t)$ si et seulement si $head^*(t') \subseteq head^*(t)$;

- si $t', t'' \in head^*(t)$ alors $t' \in head^*(t'')$ ou $t'' \in head^*(t')$;
- (corollaire) si $t' \in head^*(t)$ et $t'' \in head^*(t) \setminus head^*(t')$ alors $t' \in head^*(t'')$.

De façon analogue aux sous-types têtes, on définit ci-dessous les sous-types arguments, parmi lesquels on distingue les arguments directs et indirects :

Définition 6.6 (Sous-types arguments directs et indirects)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs orientés et t un $\mathcal{O}$-terme sur Pr. Soit $args(t)$ la fonction définie par :

- $args(t) = \emptyset$ si $t \in Pr$
- sinon, soit $t = f(t_1, \ldots, t_n)$: $args(t) = \{\, t_i \mid arg_f(i) = 1 \,\}$

$args(t)$ est l'ensemble des *arguments directs* de t.

$args^*(t)$ est la fonction définie par :

- $args^*(t) = \emptyset$ si $t \in Pr$
- sinon, $args^*(t) = args(t) \cup args^*(pr(t))$

$args^*(t)$ est l'ensemble des *arguments [indirects]* de t.

Lemme 6.2. *$t \in args^*(t'')$ si et seulement s'il existe t' tel que $t \in args(t')$ et $t' \in head^*(t'')$.*

Démonstration. $h(t'') \geq 1$, sinon t'' serait un type primitif et $args(t'')$ serait l'ensemble vide. Par induction sur la hauteur de t'' :

- $h(t'') = 1$: ($\Rightarrow$) $args^*(t'') = args(t'')$, donc $t' = t'' \in head^*(t'')$ vérifie la propriété. ($\Leftarrow$) $head^*(t'') = \{head(t''), t''\}$, donc $t' = t''$ (car $args(head(t'')) = \emptyset$), et $t \in args(t'')$ implique $t \in args^*(t'')$.
- $h(t'') > 1$.
  $\Leftarrow$. Si $t' \neq t''$, soit $t''' = pr(t'') : h(t''') < h(t'')$ et $t' \in head^*(t''')$, donc par hypothèse d'induction $t \in args^*(t''') \subseteq args^*(t'')$. Sinon $t' = t''$, ce qui implique que $t \in args(t') = args(t'') \subseteq args^*(t'')$.
  $\Rightarrow$. $t \in args^*(t'')$: si $t \in args(t'')$ alors la propriété est vraie avec $t' = t''$. Sinon $t \in args^*(pr(t))$. Or par hypothèse d'induction la propriété est vraie pour $pr(t)$, et comme $head^*(t) \subseteq head^*(pr(t))$ la propriété est aussi vraie pour t.

□

Lemme 6.3. *Si $t \in head^*(t')$ alors $args^*(t) \subseteq args^*(t')$*

Démonstration. D'après le lemme 6.2, $args^*(t) = \bigcup_{t'' \in head^*(t)} args(t'')$. Or $t \in head^*(t')$ implique $head^*(t) \in head^*(t')$, donc en appliquant le lemme 6.2 dans l'autre sens on obtient $args(t'') \subseteq args^*(t')$ pour tout $t'' \in head^*(t)$, autrement dit $args^*(t) \subseteq args^*(t'')$. □

Proposition 6.5. *Si tout type $t' \in args^*(t)$ est primitif, alors $\{\, t' \mid t'$ est un sous-type de $t \,\} = head^*(t) \cup args^*(t)$.*

Démonstration. Par induction sur $h(t)$. Si $h(t) = 0$, t est primitif donc $t' = t \in head^*(t)$. Sinon $h(t) > 0$, $t = f(t_1, \ldots, t_n)$ avec $t_h = pr(t)$: si t' est un sous-type de t_h la propriété est vraie par hypothèse d'induction (avec les lemmes 6.2 et 6.3), sinon

- soit $t' \in args(t) \subseteq args^*(t)$,
- soit $t' = t \in head^*(t)$.

□

Proposition 6.6. *Soit σ une substitution.*
Si $\sigma(head(t))$ est un type primitif alors $|head^*(\sigma(t))| = |head^*(t)|$.

Démonstration. Par induction sur $|head^*(t)|$:

- $|head^*(t)| = 1$, autrement dit t est primitif. $\sigma(head(t))$ est un type primitif, donc comme $t = head(t)\ \sigma(t)$ est aussi primitif, ainsi $|head^*(\sigma(t))| = 1$.
- $|head^*(t)| > 1$. $\sigma(head(pr(t))) = \sigma(head(t))$ est primitif et $|head^*(pr(t))| < |head^*(t)|$, donc par hypothèse d'induction $|head^*(\sigma(pr(t)))| = |head^*(pr(t))|$. Or par définition
  $$\begin{aligned} |head^*(\sigma(t))| &= |head^*(pr(\sigma(t))) \cup \{\sigma(t)\}| \\ &= |head^*(pr(\sigma(t)))| + 1, \end{aligned}$$
  et comme $pr(\sigma(t)) = \sigma(pr(t))$ (car t n'est pas primitif) :
  $$\begin{aligned} |head^*(\sigma(t))| &= |head^*(\sigma(pr(t)))| + 1 \\ &= |head^*(pr(t))| + 1 \\ &= |head^*(t) \setminus \{t\}| + 1 \\ &= |head^*(t)| \end{aligned}$$

□

Corollaire 6.2. *Si $t' \in head^*(\sigma(t))$ et $\sigma(head(t)) = head(t')$ alors $|head^*(t')| \leq |head^*(t)|$.*

Démonstration. $\sigma(head(t)) = head(t')$, donc $\sigma(head(t))$ est un type primitif. Ainsi d'après la proposition 6.6 $|head^*(\sigma(t))| = |head^*(t)|$. Or $t' \in head^*(\sigma(t))$ implique $|head^*(t')| \leq |head^*(\sigma(t))|$, d'où $|head^*(t')| \leq |head^*(t)|$. □

Proposition 6.7. *Soit σ une substitution. Si u, t, t' sont des types tels que $t \in head^*(\sigma(u))$ et $t' \in args(t)$, alors (au moins) une des deux conditions suivantes est vérifiée :*

- *il existe $u' \in args^*(u)$ tel que $\sigma(u') = t'$;*
- *ou $t \in head^*(\sigma(head(u)))$.*

Démonstration. Notons que $t'' \in args^*(t)$. On montre par induction sur la hauteur de u que l'une des conditions est toujours vraie :

- $h(u) = 0$, u est primitif. Dans ce cas $head(u) = u$, donc $t \in head^*(\sigma(head(u)))$.
- $h(u) > 0$. On distingue les deux cas ci-dessous :
  - il existe $u'' \in head^*(u)$ tel que $\sigma(u'') = t$.
    - si $u'' \neq u$ alors $h(u'') < h(u)$, donc par hypothèse d'induction la propriété est vraie pour u''. $t = \sigma(u'')$ donc $t \in head^*(\sigma(u''))$, et d'autre part $u'' \in head^*(u)$ implique que $args^*(u'') \subseteq args^*(u)$ et que $head(u'') = head(u)$, ainsi si l'une des deux propriétés est vraie pour u'' elle l'est aussi pour u.

$*$ si $u'' = u$ alors $\sigma(u) = t$. Soit $u = f(u_1, \ldots, u_n)$, avec $u_h = pr(u)$. Par définition $t = \sigma(u) = f(\sigma(u_1), \ldots, \sigma(u_n))$ (car u n'est pas primitif). Comme $t' \in args(t)$ et $args(t) = \{\sigma(u_1), \ldots, \sigma(u_{h-1}), \sigma(u_{h+1}), \ldots, \sigma(u_n)\}$, il existe $j \neq h$ tel que $t' = \sigma(u_j)$. Or $u_j \in args(u) \subseteq args^*(u)$, donc il existe $u' = u_j \in args^*(u)$ tel que $\sigma(u') = t'$.

– Sinon $\sigma(u'') \neq t$ quel que soit $u'' \in head^*(u)$. D'après les définitions de $head^*(x)$ et de la substitution, on peut facilement montrer (par induction sur la taille de $\sigma(u)$) que $head^*(\sigma(u)) = \{\sigma(u'')|u'' \in head^*(u)\} \cup head^*(\sigma(head(u)))$. Ainsi si $\sigma(u'') \neq t$ quel que soit $u'' \in head^*(u)$ alors t appartient nécessairement à $head^*(\sigma(head(u)))$.

□

Proposition 6.8. *Soit σ une substitution. Si $t' \in head^*(t)$ alors $\sigma(t') \in head^*(\sigma(t))$.*

Démonstration. Par induction sur la hauteur de t :

- $h(t) = 0$. t est primitif donc $t = t'$ implique $\sigma(t) = \sigma(t')$, par conséquent $\sigma(t') \in head^*(\sigma(t))$.
- $h(t) > 0$. Si $t = t'$ la propriété est clairement vérifiée. Sinon soit $t = f(t_1, \ldots, t_n)$ et $t_h = pr(t)$. $t \neq t'$ donc $t' \in head^*(t_h)$. Par hypothèse d'induction la propriété est vraie sur t_h, d'où $\sigma(t') \in head^*(\sigma(t_h))$. Or $\sigma(t) = f(\sigma(t_1), .., \sigma(t_h), .., \sigma(t_n))$ (avec $pr(\sigma(t)) = \sigma(t_h)$), donc $\sigma(t') \in head^*(t_h) \subseteq head^*(\sigma)(t)$.

□

Proposition 6.9. *Soit σ une substitution. Si $t' \in args^*(t)$ alors $\sigma(t') \in args^*(\sigma(t))$.*

Cette proposition se démontre de la même façon que la précédente.

6.3.2 Définition et propriétés des grammaires par consommation d'arguments

Définition 6.7 (Règles par consommation d'arguments)**.** Soit $\mathcal{O}$ un ensemble d'opérateurs orientés et $R = A_1 \ldots A_n \rightarrow B$ une règle sur $\mathcal{O}$.

A_h est un *type consommateur* pour la règle R si les conditions suivantes sont vérifiées :

- $B \in head^*(A_h)$;
- $args^*(A_h) \subseteq Var(R)$;
- Pour tout $i \neq h$, $A_i \in args^*(A_h)$;
- Si $v \in args^*(A_h)$ tel que $A_i \neq v$ pour tout i alors il n'y a qu'une seule occurrence de v dans A_h.

$R = A_1 \ldots A_n \rightarrow B$ est une *règle par consommation d'arguments* s'il existe h, $1 \leq h \leq n$, tel que A_h est un type consommateur pour R

Remarque : s'il existe, le type consommateur d'une règle est unique. En effet, supposons que A_i et A_j soient deux types consommateurs pour R : dans ce cas $A_i \in args^*(A_j)$ et $A_j \in args^*(A_i)$. Or $t \in args^*(t')$ implique que t est un sous-type strict de t', on aurait donc A_j sous-type strict de A_i et A_i sous-type strict de A_j : contradiction.

Intuitivement la définition ci-dessus impose que dans les règles par consommation d'arguments :

- Le type résultant de la dérivation est un sous-type tête du type consommateur ;
- Tous les arguments du type consommateur sont atomiques ;
- Pour chaque type de la partie gauche différent du type consommateur, il existe une occurrence de ce type comme argument du type consommateur. Cela signifie qu'aucun type ne pourra "disparaître" lors d'une dérivation sans que ce type ne corresponde à un argument d'un type consommateur. Accessoirement, cela implique que, dans la règle (mais pas nécessairement dans une instanciation de celle-ci), tous les types de la partie gauche différents du type consommateur sont atomiques.
- Les arguments du type consommateur auxquels ne correspond aucun A_i, $i \neq h$, (arguments non "consommés") ne doivent apparaître qu'une seule fois dans le type consommateur. Cette contrainte est liée au problème des types orphelins, qui fait l'objet de la partie 6.3.3 ci-dessous.

Définition 6.8 (Système de $\mathcal{R}$-grammaires par consommation d'arguments)**.** $\langle \mathcal{O}, \mathcal{R}, s \rangle$ est un *système de $\mathcal{R}$-grammaires par consommation d'arguments* si

(i) s est un opérateur atomique[8] ;

(ii) $\mathcal{O} \setminus \{s\}$ est un ensemble d'opérateurs orientés[9] ;

(iii) $\mathcal{R}$ est un ensemble de règles par consommation d'arguments sur l'ensemble d'opérateurs $\mathcal{O} \setminus \{s\}$[10]

Par extension de cette définition, On parlera de *$\mathcal{R}$-grammaire par consommation d'arguments* pour toute $\mathcal{R}$-grammaire appartenant à un tel système.

Définition 6.9 ($Heads(G)$)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une $\mathcal{R}$-grammaire par consommation d'arguments :
$Heads(G) = \{\ t \in Pr \cup \{s\} \mid \exists t' \in Lex(G)$ tel que $t = head(t')\ \}$

Définition 6.10 (Règles par consommation stricte d'arguments)**.** Soit $R = A_1 \dots A_n \rightarrow B$ une règle par consommation d'arguments, dont A_h est le type consommateur. R est une règle *par consommation stricte d'arguments* si $B \neq A_h$.

Notons qu'il n'est pas nécessaire dans une telle règle que tous les $t \in args(A_h)$ du type consommateur A_h soient "consommés" : par exemple, une règle de la forme $A/^{?}B \rightarrow A$ répond bien à tous les critères, bien que l'argument B ne soit lié à aucun autre type.

Remarque : la définition de "règles par consommation (stricte) d'arguments" est naturellement étendue aux "systèmes de $\mathcal{R}$-grammaires par consommation (stricte) d'arguments".

Proposition 6.10. *Soit $R = A_1 \dots A_n \rightarrow A_0$ une règle et $\{t_0, t_1, \dots, t_n\}$ un ensemble de types tel que $t_1 \dots t_n \rightarrow_R t_0$. Si R est une règle par consommation d'arguments alors*

(1) il existe $h > 0$ tel que $t_0 \in head^*(t_h)$;

(2) pour tout i, $i \neq h$ et $i \neq 0$, $t_i \in args^*(t_h)$;

(3) si de plus R est aussi une règle par consommation stricte d'arguments alors $t_0 \neq t_h$.

[8] donc d'arité nulle.

[9] Ces opérateurs sont donc tous d'arité non nulle.

[10] Ainsi s ne peut jamais intervenir dans les règles universelles.

Démonstration. Par définition de $\rightarrow_R$ il existe σ tel que $\sigma(A_i) = t_i$ pour tout i. R est une règle par consommation d'arguments donc il existe h ($1 \leq h \leq n$) tel que A_h est le type consommateur pour R.

(1) Par définition $A_0 \in head^*(A_h)$. Selon la proposition 6.8, cela implique que $\sigma(A_0) \in head^*(\sigma(A_h))$, autrement dit $t_0 \in head^*(t_h)$.

(2) Par définition pour tout $i \neq h$ ($1 \leq i \leq n$) $A_i \in args^*(A_h)$. Par la proposition 6.9 on obtient directement que $t_i = \sigma(A_i) \in args^*(\sigma(A_h)) = t_h$.

(3) Si R est aussi une règle par consommation stricte d'arguments alors $A_0 \neq A_h$, d'où $\sigma(A_0) \neq \sigma(A_h)$ car $head^*(\sigma(A_0)) \subset head^*(\sigma(A_h))$, autrement dit $t_0 \neq t_h$.

□

Proposition 6.11. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$. S'il existe un arbre de dérivation (partiel) P tel que $racine(P) = t$, alors il existe $t' \in Lex(G)$ et $w \in prod(P)$ tels que $w \triangleright_G t'$ et $t \in head^*(t')$.*

Démonstration. Par induction sur la hauteur de l'arbre $h(P)$.

- $h(P) = 0$. P est nécessairement de la forme $[\,](t', w)$, donc $t = t' \in head^*(t')$.
- $h(P) > 0$. $P = [R](t, P_1, \ldots, P_n)$. Soit h la position du type consommateur dans la règle R et $t'' = racine(P_h)$ le type consommateur. R est une règle par consommation d'arguments donc $t \in head^*(t'')$ (d'après la proposition 6.10). $h(P_h) < h(P)$ donc la propriété est vraie sur P_h, c'est-à-dire qu'il existe $t' \in Lex(G)$ et $w \in prod(P)$ tels que $t'' \in head^*(t')$ et $w \triangleright_G t'$. $t \in head^*(t'')$ donc t appartient aussi à $head^*(t')$ par définition.

□

Proposition 6.12. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire non vide dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*

Si G ne contient pas de types inutiles alors pour tout $t \in STLex(G)$ il existe $t' \in Lex(G)$ tel que $t \in head^*(t')$ ou $t \in args^*(t')$.

Démonstration. Supposons qu'il existe $t \in STLex(G)$ tel que $t \notin head^*(t')$ et $t \notin args^*(t')$ pour tout $t' \in Lex(G)$. G ne contient pas de type inutile donc t est utile :

Supposons que t soit un sous-type de s. $t = s$ car s est un opérateur atomique ($\langle \mathcal{O}, \mathcal{R}, s \rangle$ étant un système de $\mathcal{R}$-grammaires par consommation d'arguments). Ainsi il n'existe aucun $t' \in Lex(G)$ tel que $s \in head^*(t')$, ce qui implique selon la proposition 6.11 qu'il n'existe aucun arbre de dérivation de racine s pour G et par conséquent $SL(G) = FT(G) = \emptyset$. Or G n'est pas vide donc G contient au moins une règle $w \triangleright t'$, mais comme $FT(G) = \emptyset$ t' est nécessairement inutile. Cette contradiction implique que t n'est pas un sous-type de s.

Ainsi il existe une règle $R = A_1 \ldots A_n \rightarrow A_0 \in \mathcal{R}$, une substitution σ, un indice j et un type u tels que

- $\sigma(A_i) \in FT(G)$ pour tout i,
- u est un sous-type de A_j,

- $\sigma(u) = t$,
- et $u \in Var(R)$ si et seulement si t est primitif.

Soit h la position du type consommateur dans la règle R. On montre une contradiction dans tous les cas possibles pour la position j :

- $j \neq h$ et $j \neq 0$: R est une règle par consommation d'arguments donc $A_j \in Var(R)$, ce qui implique que $u = A_j$ donc $t = \sigma(u) = \sigma(A_j) \in FT(G)$. Par la proposition 6.11, il existe $t' \in Lex(G)$ tel que $t \in head^*(t')$: contradiction.
- $j = h$ ou $j = 0$: R est une règle par consommation d'arguments donc $args^*(A_j) \subseteq Var(R)$ (les arguments sont atomiques), ainsi la proposition 6.5 donne $u \in head^*(A_j) \cup args^*(A_j)$. D'après les propositions 6.8 et 6.9, on a donc $t = \sigma(u) \in head^*(\sigma(A_j)) \cup args^*(\sigma(A_j))$. Or $\sigma(A_j) \in FT(G)$, par conséquent selon la proposition 6.11 il existe $t' \in Lex(G)$ tel que $\sigma(A_j) \in head^*(t')$. Comme ceci implique que $head^*(\sigma(A_i)) \subseteq head^*(t')$ et $args^*(\sigma(A_i)) \subseteq args^*(t')$ (par les lemmes 6.2 et 6.3), on aboutit encore à une contradiction : $t \in args^*(t') \cup head^*(t')$.

□

Proposition 6.13. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire par consommation d'arguments sans types inutiles.*

Si G n'est pas vide alors $s \in Heads(G)$.

Démonstration. G n'est pas vide donc G contient au moins une règle $w \triangleright t$. G ne contient pas de types inutiles donc il existe un arbre de dérivation dont la racine est s contenant un nœud t. D'après la proposition 6.11, il existe $t' \in Lex(G)$ tel que $s \in head^*(t')$, or s est un type atomique donc $s \in Heads(G)$.

□

6.3.3 Arguments orphelins et grammaires ao-normales

Nous verrons dans la partie 6.4 que pour prouver l'élasticité finie des grammaires k-bornées par consommation strictes d'arguments il est nécessaire (entre autres) de pouvoir borner le nombre de types primitifs d'un certain ensemble de grammaires. C'est la raison pour laquelle il est indispensable d'étudier le cas des *arguments orphelins*, dont le rôle grammatical est a priori limité mais dont la quantité n'est pas limitée.

Cette partie a donc pour unique objet de définir les arguments orphelins et de montrer qu'on peut en borner la quantité sans modifier le langage de la grammaire, à travers une sorte de normalisation nommée "grammaires ao-normales".

Définition 6.11 (Types orphelins)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$. Un type $t \in STLex(G)$ est un *type orphelin* dans G si $t \notin head^*(t')$ pour tout $t' \in Lex(G)$.

L'ensemble des types orphelins d'une grammaire G est noté $TO(G)$.

Corollaire 6.3 (Arguments orphelins)**.** *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$. Si $t \in STLex(G)$ est un type orphelin dans G alors il existe $t' \in Lex(G)$ tel que $t \in args^*(t')$.*

Démonstration. $t \notin head^*(t')$ quel que soit $t' \in Lex(G)$ et G ne contient pas de types inutiles, donc la proposition 6.12 impose qu'il existe $t' \in Lex(G)$ tel que $t \in args^*(t')$. □

Comme on s'intéressera uniquement aux grammaires sans types inutiles, on parlera *d'arguments* orphelins plutôt que de *types* orphelins. Intuitivement, un argument orphelin est un type qui n'apparaît qu'en position d'argument dans la grammaire, ce qui signifie que cet argument ne sera jamais consommé car il faudrait pour cela qu'il existe au moins une occurrence de ce type en tant que sous-type tête. La seule utilisation possible de cet argument est donc une simple élimination.

Nous proposons ci-dessous une normalisation des grammaires permettant de borner le nombre de types orphelins. Le principal problème est de s'assurer l'équivalence des langages entre la grammaire d'origine et sa forme normalisée. En effet, on pourrait supposer que, puisque les arguments orphelins ne sont jamais consommés, il suffit de les remplacer tous par un unique type spécial. Mais le problème est plus complexe que cela, car même les arguments orphelins peuvent jouer un rôle de par leur position au niveau de l'unification. Il est donc indispensable de garantir que la transformation proposée ne permet pas de réaliser des unifications qui n'auraient pas pu avoir lieu dans la grammaire d'origine. L'exemple ci-dessous illustre la différence de langages causée par un remplacement direct de tous les arguments orphelins par un type unique.

Exemple 6.1. Soit $\mathcal{R} = \{\ A/B\ B \to A\ ;\ A/^?B\ B \to A\ ;\ A/^?B \to A\ \}$.

Dans la grammaire G ci-dessous, les types c et d sont orphelins. σ est la substitution $\{\ c \mapsto t_{ao}; d \mapsto t_{ao}\ \}$, qui remplace tout type orphelin par un unique type t_{ao}.

$$G = \begin{cases} x : s/(b/^?c) \\ y : s/(b/^?d) \\ z : b/^?c \\ w : b/^?d \\ u : s/b \end{cases} \quad \rightarrow \quad \sigma[G] = \begin{cases} x : s/(b/^?t_{ao}) \\ y : s/(b/^?t_{ao}) \\ z : b/^?t_{ao} \\ w : b/^?t_{ao} \\ u : s/b \end{cases}$$

$$L(G) = \{xz, yw, uz, uw\} \qquad L(\sigma[G]) = \{xz, yw, xw, yz, uz, uw\}$$

On constate que l'unification des arguments orphelins c et d autorise des dérivations impossibles dans la grammaire d'origine, il faut donc bien sûr éviter ce type de généralisation dans la normalisation.

Proposition 6.14. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*

Si $t \in STLex(G)$ est un argument orphelin, alors t est un type primitif.

Démonstration. Soit t un argument orphelin utile dans G. Supposons que $t \notin Pr$, c'est-à-dire $t = f(t_1, \ldots, t_n)$. t est utile donc il existe une règle $R = A_1 \ldots A_n \to A_0 \in \mathcal{R}$, une substitution σ, un indice j et un type u tels que

- $\sigma(A_i) \in FT(G)$ pour tout i,
- u est un sous-type de A_j,
- $\sigma(u) = t$,
- et $u = f(u_1, \ldots, u_n)$ (u ne peut être primitif puisque $u \in Var(R)$ si et seulement si t est primitif).

Soit h la position du type consommateur dans la règle R.

- Si $j \neq h$ et $j \neq 0$ alors $A_j \in Var(R)$, or A_j contient le type complexe $u = f(u_1, \ldots, u_n)$: impossible.
- Si $j = h$ ou $j = 0$, alors $u \in head^*(A_j) \cup args^*(A_j)$ (par la proposition 6.5), donc $t = \sigma(u) \in head^*(\sigma(A_j))$ (proposition 6.8). Or $\sigma(A_j) \in FT(G)$, donc d'après la proposition 6.11 il existe $t' \in Lex(G)$ tel que $t \in head^*(\sigma(A_j)) \subseteq head^*(t')$. Ceci contredit le fait que t soit un type orphelin.

□

Corollaire 6.4. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*

Si $t \in STLex(G)$ est un type complexe alors il existe $t' \in Lex(G)$ tel que $t \in head^*(t')$.

Démonstration. Par la contraposée de la proposition 6.14 et la définition 6.11 des types orphelins. □

Définition 6.12 (Grammaire ao-normale)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$. G est une *grammaire ao-normale* si

$$|\{\, t \in STLex(G) \mid t \in TO(G) \,\}| \leq C_{ao} \ , \quad \text{avec } C_{ao} = |G|^2 + 1.$$

Remarque : Comme on le verra par la suite, la définition des grammaires ao-normales a uniquement pour but de borner le nombre d'arguments orphelins présents dans la grammaire. La constante $C_{ao} = |G|^2 + 1$ est utilisée ici car nous montrerons que toute grammaire (par consommation d'arguments) peut être convertie en une grammaire ayant au maximum ce nombre d'arguments orphelins. Néanmoins toute autre constante conviendrait, et il est d'ailleurs probable que celle-ci ne soit pas optimale.

Définition 6.13 (*AODiff*(t, t'))**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.

Étant donné deux types $t, t' \in STLex(G)$ on définit la fonction *AODiff*(t, t') de la façon suivante :

(i) Si $t = t'$ alors *AODiff*$(t, t') = \emptyset$. Sinon, soit $\{u_1, \ldots, u_n\} = head^*(t)$ (resp. $\{u'_1, \ldots, u'_{n'}\} = head^*(t')$), avec $i \leq j$ si et seulement si $u_i \in head^*(u_j)$ (resp. $i \leq j$ ssi $u'_i \in head^*(u'_j)$) (de telles séquences existent par définition) ;

(ii) Si $\{u_1, \ldots, u_n\} \subseteq \{u'_1, \ldots, u'_{n'}\}$ ou $\{u'_1, \ldots, u'_{n'}\} \subseteq \{u_1, \ldots, u_n\}$, alors *AODiff*$(t, t') = \emptyset$. Sinon (il existe i tel que $u_i \neq u'_i$), soit $i = \min(\{\, 1 \leq j \leq \min(n, n') \mid u_j \neq u'_j \,\})$;

(iii) Si $i = 1$ alors *AODiff*$(t, t') = \emptyset$. Sinon (u_i et u'_i ne sont pas primitifs), soit $u_i = f(t_1, \ldots, t_m)$ et $u'_i = f'(t'_1, \ldots, t'_{m'})$;

(iv) Si $f \neq f'$ ou $m \neq m'$ alors *AODiff*$(t, t') = \emptyset$. Sinon ($f = g$ et $m = m'$), il existe j tel que $t_j \neq t'_j$;

(v) S'il existe $t_j \neq t'_j$ tels que t_j ou t'_j n'est pas un argument orphelin alors *AODiff*$(t, t') = \emptyset$. Sinon soit $j = \min(\{\, 1 \leq k \leq m \mid t_k \neq t'_k \,\})$ (t_j et t'_j sont des arguments orphelins) : *AODiff*$(t, t') = \{t_j, t'_j\}$.

Remarque : l'indice i peut être défini comme l'unique indice tel que $u_i \in head^*(t)$ et $u'_i \in head^*(t')$ avec $pr(u_i) = pr(u'_i)$ mais $u_i \neq u'_i$.

Le principe de la fonction ***AODiff***(t, t') est d'assurer une distinction minimale entre deux types, en particulier lorsque ceux-ci ne se distinguent que par des sous-types orphelins. Ainsi, dans la plupart des cas la fonction renvoie l'ensemble vide : il suffit que t et t' soient différenciables d'une manière indépendante des types orphelins, ou qu'ils soient identiques. S'il n'existe aucune autre distinction possible, la fonction renvoie un couple de types orphelins qui distinguent t et t'. Il faut noter que la façon dont cette fonction est construite permet de choisir le "premier" couple de types différents rencontrés, ce qui est nécessaire pour que ce couple de types soit suffisant pour distinguer (si besoin) n'importe quel couple de types pris dans $head^*(t) \times head^*(t')$.

Définition 6.14 (σ_{ao} : normalisation des arguments orphelins)**.** Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$. Soit

$$\textit{AODiff}(G) = \bigcup_{(t,t') \in (Lex(G))^2} \textit{AODiff}(t, t'),$$

et t_{ao} un nouveau type primitif. Pour tout $t \in Var(G)$ la substitution σ_{ao} est définie par

$$\sigma_{ao}(t) = \begin{cases} t & \text{si } t \in \textit{AODiff}(G) \cup Heads(G), \\ t_{ao} & \text{sinon.} \end{cases}$$

Il faut noter que tout type primitif est toujours remplacé par un type primitif avec la substitution σ_{ao}, donc $\sigma_{ao}(t)$ est toujours isomorphe à t. Ceci a pour conséquence immédiate que s'il existe un sous-type $t \in STLex(\sigma_{ao}[G])$ alors il existe (au moins) un sous-type $t' \in STLex(G)$ tel que $\sigma_{ao}(t') = t$. On notera aussi que par définition si t est un type utile dans G alors $\sigma_{ao}(t)$ est un type utile dans $\sigma_{ao}[G]$. Enfin remarquons que t est orphelin dans G si et seulement si $\sigma_{ao}(t)$ est orphelin dans $\sigma_{ao}[G]$.

Proposition 6.15. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$, et $t, t' \in STLex(G)$.*

Si t n'est pas un argument orphelin et $t \neq t'$ alors $\sigma_{ao}(t) \neq \sigma_{ao}(t')$.

Démonstration. Notons d'abord que $\sigma_{ao}(t)$ est toujours isomorphe à t, car chaque variable y est remplacée par une variable. Donc si t et t' ne sont pas isomorphes alors il est clair que $\sigma_{ao}(t) \neq \sigma_{ao}(t')$. Nous traiterons donc uniquement du cas où t et t' sont isomorphes. On montre par induction sur $h(t) = h(t')$ que si t n'est pas orphelin et $t \neq t'$ alors $\sigma_{ao}(t) \neq \sigma_{ao}(t')$:

- t et t' sont primitifs : t n'est pas orphelin donc il existe $t'' \in Lex(G)$ tel que $t \in head^*(t'')$, autrement dit $t \in Heads(G)$:
  - si t' n'est pas orphelin ($t' \in Heads(G)$) ou si $t' \in \textit{AODiff}(G)$, alors $\sigma_{ao}(t) = t \neq t' = \sigma_{ao}(t')$;
  - sinon $t' \notin \textit{AODiff}(G)$ est orphelin, donc $\sigma_{ao}(t') = t_{ao} \neq t = \sigma_{ao}(t)$.
- $h(t) = h(t') > 0$. Soit $t = f(t_1, \ldots, t_n)$ et $t' = f(t'_1, \ldots, t'_n)$ (t et t' sont isomorphes). Il existe $t_i \neq t'_i$ puisque $t \neq t'$:

Cas 1. Il existe t_j et t'_j, $t_j \neq t'_j$, tels que l'un de ces deux types n'est pas un argument orphelin. Alors par hypothèse d'induction $t_j \neq t'_j$ implique $\sigma_{ao}(t_j) \neq \sigma_{ao}(t'_j)$, car $h(t_j) = h(t'_j) < h(t) = h(t')$. Ainsi
$\sigma_{ao}(t) = f(\sigma_{ao}(t_1), \ldots, \sigma_{ao}(t_j), \ldots, \sigma_{ao}(t_n)) \neq f(\sigma_{ao}(t'_1), \ldots, \sigma_{ao}(t'_j), \ldots, \sigma_{ao}(t'_n)) = \sigma_{ao}(t')$.

Cas 2. Sinon tous les couples (t_i, t'_i) tels que $t_i \neq t'_i$ sont des couples d'arguments orphelins. t et t' sont des types complexes donc d'après le corollaire 6.4 il existe $u, u' \in Lex(G)$ tels que $t \in head^*(u)$ et $t' \in head^*(u')$. Soit $t_h = pr(t)$ et $t'_h = pr(t')$: par définition $t_h \in head^*(t) \subseteq head^*(u)$ et $t'_h \in head^*(t') \subseteq head^*(u')$. Ainsi t_h et t'_h ne sont pas des arguments orphelins, donc $t_h = t'_h$ (sinon on serait dans le cas 1).
Soit $j = \min(\{\ 1 \leq k \leq n \mid t_k \neq t'_k\ \})$. Par définition *AODiff*(u, u') = *AODiff*$(t, t') = \{t_j, t'_j\}$ car $t_h = t'_h$ et $t \neq t'$. Ainsi $\{t_j, t'_j\} \in$ *AODiff*(G), par conséquent $\sigma_{ao}(t_j) = t_j \neq t'_j = \sigma_{ao}(t'_j)$. □

Corollaire 6.5. *Soit G une $\mathcal{R}$-grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*

σ_{ao} définit une bijection entre les ensembles

$$\{\ t \in STLex(G) \mid t \notin TO(G)\ \} \text{ et } \{\ t' \in STLex(\sigma_{ao}[G]) \mid t' \notin TO(\sigma_{ao}[G])\ \}.$$

Démonstration. Le fait que σ_{ao} est injective (sur les ensembles de types non orphelins) découle directement de la proposition 6.15.

σ_{ao} est surjective : En effet, σ_{ao} associe toujours un type primitif à un type primitif donc chaque sous-type d'un type $t' \in Lex(\sigma_{ao}[G])$ est nécessairement l'image du sous-type correspondant dans $t \in Lex(G)$, avec t l'antécédent de t' par σ_{ao} (voir la définition d'une substitution). Ainsi toute image dans $\{t' \in STLex(\sigma_{ao}[G]) | t' \notin TO(\sigma_{ao}[G])\}$ a au moins un antécédent dans $\{t \in STLex(G) | t \notin TO(G)\}$. □

Dans la suite on notera $\sigma_{ao}^{-1}(t)$ l'unique type t' tel que $\sigma_{ao}(t') = t$, pour tout t non orphelin.

Proposition 6.16. *Soit G une $\mathcal{R}$-grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*

S'il existe un arbre de dérivation P dans $\sigma_{ao}[G]$ alors il existe un unique arbre P'dans G, isomorphe à P, tel que pour tout nœud de P' dont le type est t' le nœud correspondant dans P est $t = \sigma_{ao}(t')$, et $prod(P') = prod(P)$.

Démonstration. Par induction sur la hauteur $h(P)$:

- $h(P) = 0$. $P = [](t, w)$ avec $t \in Lex(\sigma_{ao}[G])$ et $w \triangleright_{\sigma_{ao}[G]} t$. Par définition il existe $w \triangleright_G t'$ tel que $t = \sigma_{ao}(t')$, et par le corollaire 6.5 t est unique (car t' n'est pas orphelin). $prod(P') = prod(P) = \langle w \rangle$.
- $h(P) > 0$. Soit $P = [R](t_0, P_1, \ldots, P_n)$ (avec $t_0 = t$). Par hypothèse d'induction les arbres de dérivations $P'_1, \ldots, P'_n$ tels que tout nœud t' dans P'_i corresponde au nœud $\sigma_{ao}(t')$ dans P_i existent et sont (chacun) uniques. Soit $racine(P_i) = t_i$ et $racine(P'_i) = t'_i$ pour tout i, on a

donc $\sigma_{ao}(t'_i) = t_i$ pour tout i. Soit $R = A_1 \dots A_n \to A_0$, par définition il existe une substitution $\sigma : Var(R) \mapsto STLex(\sigma_{ao}[G])$ telle que $\sigma(A_i) = t_i$ pour tout i. $t'_0 = \sigma_{ao}^{-1}(t_0)$ existe et est unique d'après le corollaire 6.5, car t_0 ne peut être orphelin.

Montrons que $t'_1 \dots t'_n \to_R t'_0$, c'est-à-dire qu'il existe une substitution $\sigma' : Var(R) \mapsto STLex(G)$ telle que $\sigma'(A_i) = t'_i$ pour tout $i \geq 0$. Pour tout $v \in Var(R)$, on définit $\sigma'(v)$ de la façon suivante :

– si $\sigma(v)$ n'est pas un argument orphelin, soit $\sigma'(v) = \sigma_{ao}^{-1}(\sigma(v))$: $\sigma'(v)$ est unique par le corollaire 6.5, et cette unicité associée à la définition d'une substitution implique que pour chaque occurrence de $\sigma(v)$ dans un t_i, la position correspondante dans t'_i est nécessairement occupée par $\sigma'(v)$.
– si $\sigma(v)$ est orphelin. Soit h la position du type consommateur dans R. Le fait que R soit une règle par consommation d'arguments a pour conséquence que si $i \neq h$ et $i \neq 0$ alors $A_i \in Var(R)$. Ainsi d'après la proposition 6.11 v ne peut apparaître que dans A_h et/ou A_0. De plus $A_0 \in head^*(A_h)$ et $v \neq head(A_h)$, donc $v \in args^*(A_h)$ et il n'y a qu'une seule occurrence de v dans A_h (toujours parce que R est une règle par consommation d'arguments). Soit $\sigma'(v) = u$, avec u l'unique type dans t'_h occupant la même position que $\sigma(v)$ dans t_h : $\sigma(v) = \sigma_{ao}(u) = \sigma_{ao}(\sigma'(v))$[11]. La seule autre occurrence possible de v dans la règle R se trouve dans A_0. Comme $t_0 \in head^*(t_h)$ on a nécessairement $t'_0 \in head^*(t'_h)$ (par le corollaire 6.5), donc dans t'_0 l'occurrence correspondant à celle de v dans A_0 est bien occupée par $u = \sigma'(v)$.

On obtient donc une substitution σ' bien définie et vérifiant $\sigma'(A_i) = t'_i$ pour tout i, autrement dit $t'_1 \dots t'_n \to_R t'_0$, avec $t' = t'_0$ l'unique type dans G tel que $\sigma_{ao}(t') = t_0 = t$. Par conséquent l'arbre de dérivation $P' = [R](t', P'_1, \dots, P'_n)$ existe de manière unique et vérifie la propriété selon laquelle chaque nœud t' dans P' correspond à un nœud $\sigma_{ao}(t')$ dans P. Ce nœud de P utilise exactement la même règle que le nœud correspondant de P', donc les deux arbres sont totalement isomorphes. De plus par hypothèse d'induction $prod(P_i) = prod(P'_i)$ pour tout i, or par définition $prod(P') = prod(P'_1) \bullet \dots \bullet prod(P'_n) = prod(P_1) \bullet \dots \bullet prod(P_n) = prod(P)$. □

Corollaire 6.6. *Si G est une $\mathcal{R}$-grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$, alors $SL(G) = SL(\sigma_{ao}[G])$.*

Démonstration. $SL(G) \subseteq SL(\sigma_{ao}[G])$ d'après la proposition 4.11.

Soit $S \in SL(\sigma_{ao}[G])$: il existe un arbre de dérivation complet P qui est une instance de S. Selon la proposition 6.16, il existe un arbre P' dans G dont le produit est identique à celui de P et P' est totalement isomorphe à P. En d'autres termes, P' est aussi une instance de S, ce qui implique que $S \in SL(G)$. Par conséquent $SL(\sigma_{ao}[G]) \subseteq SL(G)$, donc $SL(G) = SL(\sigma_{ao}[G])$. □

Proposition 6.17. *Si G est une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$, alors $\sigma_{ao}[G]$ est une grammaire ao-normale sans types inutiles.*

[11] Mais notons qu'il peut y avoir plusieurs types t'' tels que $\sigma(v) = \sigma_{ao}(t'')$.

Démonstration. Soit G une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$. Le fait que $\sigma_{ao}[G]$ soit une grammaire sans types inutiles est trivial compte tenu du corollaire 6.5 et de la proposition 6.16 (en effet on peut aisément démontrer que si t est inutile dans $\sigma_{ao}[G]$ alors $\sigma_{ao}^{-1}(t)$ est inutile dans G).

$|Lex(G)| = |G|$ donc $|\{ (t,t') \in (Lex(G))^2 \}| \leq (|G|)^2$. Par définition $\mathit{AODiff}(t,t') \leq 2$ pour tout couple (t,t'), et $\mathit{AODiff}(t,t') = \mathit{AODiff}(t',t)$.

$$\mathit{AODiff}(G) = \bigcup_{(t,t') \in (Lex(G))^2} \mathit{AODiff}(t,t'),$$

donc $|\mathit{AODiff}(G)| \leq |G|^2$.

Soit $t, t' \in STLex(G)$ et $u, u' \in STLex(\sigma_{ao}[G])$ des types tels que $u = \sigma_{ao}(t)$ et $u' = \sigma_{ao}(t')$: on peut facilement montrer que $t \in head^*(t')$ et $t' \in Lex(G)$ si et seulement si $u \in head^*(u')$ et $u' \in Lex(\sigma_{ao}[G])$, autrement dit t est orphelin dans G si et seulement si $\sigma_{ao}(t)$ est orphelin dans $\sigma_{ao}[G]$. La substitution σ_{ao} est définie par la substitution identité pour tous les arguments orphelins appartenant à $\mathit{AODiff}(G)$, et par $\sigma_{ao}(t) = t_{ao}$ pour tout autre argument orphelin t. Ainsi

$$\begin{aligned} |\{ t \in STLex(\sigma_{ao}[G]) \mid t \in TO(\sigma_{ao}[G]) \}| &= |\mathit{AODiff}(G) \cup \{t_{ao}\}| \\ &\leq |G|^2 + 1. \end{aligned}$$

□

Corollaire 6.7. *Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires par consommation d'arguments.*

Pour toute $\mathcal{R}$-grammaire G dans ce système il existe une $\mathcal{R}$-grammaire G' ao-normale sans types inutiles telle que $SL(G') = SL(G)$ et $|G'| \leq |G|$.

Démonstration. $\sqsubseteq$ est une relation de réduction dans le système de grammaires $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}^\mathcal{R}, SL \rangle$ (avec $\mathcal{G}^\mathcal{R}$ l'ensemble des $\mathcal{R}$-grammaires), donc d'après la proposition 5.3 il existe une grammaire $G'' \sqsubseteq G$ faiblement $\sqsubseteq$-réduite telle que $SL(G'') = SL(G)$. G'' ne contient pas de types inutiles car elle est faiblement réduite (par la proposition 6.1). Ainsi on peut appliquer la proposition 6.17, qui donne que la grammaire $G' = \sigma_{ao}[G'']$ est ao-normale et ne contient pas de types inutiles. Le corollaire 6.6 indique que $SL(G') = SL(G'') = SL(G)$. De plus $G' = \sigma_{ao}[G'']$ donc $|G'| = |G''|$, et comme $G'' \sqsubseteq G$ implique $|G''| \leq |G|$ (proposition 4.14), on a bien $|G'| \leq |G|$. □

Proposition 6.18. *Si G est une $\mathcal{R}$-grammaire non vide sans types inutiles, alors $Var(G) \cup \{s\} = Heads(G) \cup TO(G)$.*

Démonstration. Tout $t \in Heads(G)$ est un type primitif par définition, et selon la proposition 6.14 tout argument orphelin est aussi primitif. Ainsi il est clair que $Heads(G) \cup TO(G) \subseteq Var(G) \cup \{s\}$.

Selon la proposition 6.13, s appartient à $Heads(G)$ car G n'est pas vide. Soit $t \in Var(G)$: $t \notin Heads(G)$ signifie (par définition) que $t \notin head^*(t')$ quel que soit $t' \in Lex(G)$, ce qui correspond à la définition d'un type orphelin. Ainsi pour tout $t \in Var(G)$ si $t \notin Heads(G)$ alors $t \in TO(G)$, donc on a bien aussi $Var(G) \subseteq Heads(G) \cup TO(G)$.

Par conséquent on a montré que $Var(G) \cup \{s\} \subseteq Heads(G) \cup TO(G)$ et que $Heads(G) \cup TO(G) \subseteq Var(G) \cup \{s\}$, autrement dit $Var(G) \cup \{s\} = Heads(G) \cup TO(G)$.

□

Cette dernière proposition est importante car elle permet de considérer séparément le cas des types primitifs de $Heads(G)$ de celui des types orphelins de $TO(G)$: comme ces derniers sont en nombre limité dans les grammaires ao-normales, cela permet d'étudier indépendamment les types de $Heads(G)$, comme on le verra dans la partie suivante.

6.4 Élasticité finie des $\mathcal{R}$-grammaires par consommation stricte d'arguments

Nous abordons ici la partie principale de la démonstration concernant l'élasticité finie des grammaires par consommation stricte d'arguments. Cette démonstration est en partie inspirée de celle de KANAZAWA pour les grammaires AB, mais le niveau de généralisation que nous proposons la rend bien sûr plus complexe. Le but est de parvenir à borner dans tous les cas le nombre de grammaires d'une séquence de la forme $G_1 \sqsubset G_2 \sqsubset \cdots \sqsubset G_n$ (modulo langages de structures identiques). Pour atteindre cet objectif, nous montrerons qu'il est impossible d'appliquer indéfiniment des substitutions sans créer de types inutiles.

6.4.1 Propriétés des grammaires par consommation stricte d'arguments

Dans cette partie[12] nous montrerons les différentes propriétés qui permettent de borner le nombre d'éléments dans une séquence de grammaires obtenues par substitutions successives, dans le cas où elles ont toute le même nombre de types primitifs têtes ($Heads(G)$). En effet il est relativement facile de voir que $|Heads(\sigma[G])| \leq |Heads(G)|$ (sous les conditions détaillées ci-dessous). Comme on s'intéressera seulement aux grammaires k-bornées, $|Heads(G)|$ sera borné : on voit bien alors qu'il faut montrer l'existence d'une borne sur le nombre de grammaires G dans la séquence telles que $Heads(G)$ est constant, afin de garantir que la séquence globale est finie.

Lemme 6.4. *Soit $\mathcal{R}$ un ensemble de règles par consommation d'arguments, $G = \langle \Sigma, Pr_G, \triangleright_G \rangle$ une $\mathcal{R}$-grammaire sans types inutiles et σ une substitution fidèle telle que*

- *$G' = \sigma[G] = \langle \Sigma, Pr_{G'}, \triangleright_{G'} \rangle$ ne contient pas de types inutiles ;*
- *$|Heads(G)| = |Heads(G')|$.*

Quels que soient $x, x' \in Heads(G)$: si $x \neq x'$ alors $head(\sigma(x)) \neq head(\sigma(x'))$.

Démonstration. Notons d'abord que

[12] Techniquement, cette partie de la démonstration trouve son équivalent dans le chapitre 5 de [60]. En particulier, le lemme clé (*Key lemma*) 5.56 (p. 76) de KANAZAWA, qui joue un rôle primordial dans la démonstration de l'élasticité finie des grammaires AB rigides, correspond approximativement dans notre généralisation à la proposition 6.20 (à laquelle on peut ajouter les conséquences qui la suivent).

$$\begin{aligned} Heads(G') &= Heads(\sigma[G]) \\ &= \{\, head(t) \mid t \in Lex(\sigma[G]) \,\} \\ &= \{\, head(\sigma(t)) \mid t \in Lex(G) \,\} \\ &= \{\, head(\sigma(head(t))) \mid t \in Lex(G) \,\} \\ &= \{\, head(\sigma(t)) \mid t \in Heads(G) \,\}. \end{aligned}$$

Supposons qu'il existe $x \neq x' \in Heads(G)$ tels que $head(\sigma(x)) = head(\sigma(x'))$: il est clair que cela implique $|Heads(G)| > |\{\, head(\sigma(t)) \mid t \in Heads(G) \,\}| = |Heads(G')|$, d'où contradiction avec l'hypothèse $|Heads(G)| = |Heads(G')|$. □

Lemme 6.5. *Soit $\mathcal{R}$ un ensemble de règles par consommation d'arguments, $G = \langle \Sigma, Pr_G, \triangleright_G \rangle$ une $\mathcal{R}$-grammaire sans types inutiles et σ une substitution fidèle telle que*

- *$G' = \sigma[G] = \langle \Sigma, Pr_{G'}, \triangleright_{G'} \rangle$ ne contient pas de types inutiles ;*
- *$|Heads(G)| = |Heads(G')|$.*

Soit $x \in Heads(G)$ et $t \in Lex(\sigma[G])$.
S'il existe $x' \in head^*(\sigma(x))$ tel que $x' \in head^*(t)$, alors $\sigma(x) \in head^*(t)$.

Démonstration. Soit $t \in Lex(\sigma[G])$ tel que $x' \in head^*(t)$, avec $x' \in head^*(\sigma(x))$. Soit $u \in Lex(G)$ le type tel que $t = \sigma(u)$, et $y = head(u)$. On peut déduire directement des hypothèses que $head(\sigma(y)) = head(\sigma(u)) = head(t) = head(x') = head(\sigma(x))$, d'où $y = x$ d'après le lemme 6.4. Or selon la proposition 6.8 $x \in head^*(u)$ implique $\sigma(x) \in head^*(\sigma(u))$, autrement dit $\sigma(x) \in head^*(t)$.

□

Remarque : comme nous utilisons souvent la formule $head^*(pr(\sigma(x)))$ dans la suite, rappelons que par définition $head^*(pr(x)) = head^*(x) \setminus \{x\}$. Par ailleurs cette formule n'a de sens que si x est un type complexe.

Proposition 6.19. *Soit $\mathcal{R}$ un ensemble de règles par consommation d'arguments, $G = \langle \Sigma, Pr_G, \triangleright_G \rangle$ une $\mathcal{R}$-grammaire sans types inutiles et σ une substitution fidèle telle que*

- *$G' = \sigma[G] = \langle \Sigma, Pr_{G'}, \triangleright_{G'} \rangle$ ne contient pas de types inutiles ;*
- *$|Heads(G)| = |Heads(G')|$.*

Pour tout $x \in Heads(G)$, si $x' \in head^*(pr(\sigma(x)))$ alors $x' \neq s$ et $x' \neq \sigma(t)$ pour tout $t \in STLex(G)$.

Démonstration. Soit $x \in Heads(G)$ et $x' \in head^*(pr(\sigma(x)))$. $\sigma(x)$ est donc un type complexe. On montre par l'absurde que $x' \neq s$ et $x' \neq \sigma(t)$ pour tout $t \in STLex(G)$:

- Supposons que $x' = s$. Dans ce cas $s = head(\sigma(x))$ car s est atomique. Or $\sigma(x)$ est un type complexe donc $x \neq s$ car $\sigma(s) = s$ (par définition d'une substitution). Par conséquent le lemme 6.4 donne $head(\sigma(x)) \neq head(\sigma(s)) = s$, d'où $head(x') \neq s$ et bien sûr $x' \neq s$.
- Supposons qu'il existe $t \in STLex(G)$ tel que $x' = \sigma(t)$.
  - si $head(t) = x$, alors $\sigma(x) \in head^*(\sigma(t))$, ce qui implique que $\sigma(t) \notin head^*(pr(\sigma(x)))$, donc $\sigma(t) \neq x'$.

– si $head(t) = y \neq x : \sigma(y) \in head^*(\sigma(t))$ selon la proposition 6.8, or $y \neq x$ implique $head(\sigma(y)) \neq head(\sigma(x))$ d'après le lemme 6.4. Ainsi on a $head(\sigma(t)) = head(\sigma(y))$ et $head(\sigma(y)) \neq head(\sigma(x)) = head(x')$, donc $x' \neq \sigma(t)$.

□

Proposition 6.20. *Soit $\mathcal{R}$ un ensemble de règles par consommation stricte d'arguments, $G = \langle \Sigma, Pr_G, \triangleright_G \rangle$ une $\mathcal{R}$-grammaire sans types inutiles et σ une substitution fidèle telle que la $\mathcal{R}$-grammaire $G' = \sigma[G] = \langle \Sigma, Pr_{G'}, \triangleright_{G'} \rangle$ ne contient pas de types inutiles et $|Heads(G)| = |Heads(G')|$.*

Soit P un arbre de dérivation (partiel) pour G'.

S'il existe $x \in Heads(G)$ tel qu'il existe dans P un nœud de type x', avec $x' \in head^*(pr(\sigma(x)))$, alors il existe un type $y \in Heads(G)$ tel que $racine(P) \in head^*(pr(\sigma(y)))$.

Démonstration. $\sigma(x)$ est un type complexe car il existe $x' \in head^*(pr(\sigma(x)))$. Soit P' le sous-arbre de P dont la racine est x'. On montre la propriété par induction sur la différence de hauteur $d = h(P) - h(P')$ de ces deux arbres :

- $d = 0$, les deux arbres sont identiques. Dans ce cas $racine(P) = racine(P') = x'$, ainsi avec $y = x'$ la propriété est facilement vérifiée.
- $d > 0$. Soit $P = [R](t_0, P_1, \ldots, P_n)$, et soit $t_i = racine(P_i)$ pour tout $i > 0$. Soit P_j le sous-arbre contenant P'. $h(P_j) < h(P)$ d'où $h(P_j) - h(P') < d$, donc par hypothèse d'induction la propriété est vraie pour P_j, c'est-à-dire qu'il existe $z \in Heads(G)$ tel que $t_j = racine(P_j) \in head^*(pr(\sigma(z)))$ ($\sigma(z)$ est donc un type complexe). R est une règle par consommation stricte d'arguments. Selon la forme de R et la position de t_j dans l'application de cette règle, on distingue les deux cas suivants :
  – si t_j correspond au type consommateur pour R, alors par définition $t_0 \in head^*(t_j)$ (proposition 6.10). Or $t_j \in head^*(pr(\sigma(z)))$ donc $t_0 \in head^*(pr(\sigma(z)))$: ainsi il existe $y = z \in Heads(G)$ tel que $t_0 = racine(P) \in head^*(pr(\sigma(y)))$.
  – sinon, t_j correspond à un type consommé dans la règle R. Soit h la position du type consommateur ($h \neq j$) : par définition t_h est tel que $t_j \in args^*(t_h)$ (proposition 6.10). D'après la proposition 6.11, il existe $w \in prod(P)$ tel que $w \triangleright_{G'} t$ et $t_h \in head^*(t)$, donc $t_j \in args^*(t)$ (par application du lemme 6.2). Soit $u \in Lex(G)$ le type tel que $\sigma(u) = t$ (u existe puisque $G' = \sigma[G]$). $t_j \in head^*(pr(\sigma(z)))$, avec $z \in Heads(G)$, ce qui implique selon la proposition 6.19 que $\sigma(u') \neq t_j$ pour tout $u' \in STLex(G)$, et en particulier pour tout $u' \in args^*(u)$. Par conséquent il découle de la proposition 6.7 que $t_h \in head^*(\sigma(head(u)))$. Soit $y = head(u)$. Comme R est une règle par consommation stricte d'arguments, $t_0 \in head^*(t_h)$ est un sous-type tête strict de t_h (proposition 6.10), autrement dit $t_0 \in head^*(pr(t_h))$. $t_h \in head^*(\sigma(y))$, donc $t_0 \in head^*(pr(\sigma(y)))$. Ainsi il existe $y \in Heads(G)$ tel que $t_0 = racine(P) \in head^*(pr(\sigma(y)))$.

□

Corollaire 6.8. *Soit $\mathcal{R}$ un ensemble de règles par consommation stricte d'arguments, $G = \langle \Sigma, Pr_G, \triangleright_G \rangle$ une $\mathcal{R}$-grammaire sans types inutiles et σ une substitution fidèle telle que*

- *$G' = \sigma[G]$ ne contient pas de types inutiles ;*
- *$|Heads(G)| = |Heads(G')|$.*

Pour tout $x \in Heads(G)$, s'il existe $x' \in head^*(pr(\sigma(x)))$, alors $x' \notin FT(\sigma[G])$.

Démonstration. Soit $x' \in head^*(pr(\sigma(x)))$, supposons que $x' \in FT(\sigma[G])$: ainsi il existe un arbre de dérivation complet P pour $\sigma[G]$ (donc $racine(P) = s$) contenant un nœud de type x', avec $x' \in head^*(pr(\sigma(x)))$. D'après la proposition 6.20, cela implique qu'il existe un type $y \in Heads(G)$ tel que $racine(P) \in head^*(pr(\sigma(y)))$. D'après la proposition 6.19 cela implique que $racine(P) \neq s$, ce qui constitue une contradiction puisque P est par hypothèse un arbre de dérivation complet. □

Proposition 6.21. *Soit $\mathcal{R}$ un ensemble de règles par consommation stricte d'arguments, $G = \langle \Sigma, Pr_G, \triangleright_G \rangle$ une $\mathcal{R}$-grammaire sans types inutiles et σ une substitution fidèle telle que*

- *$G' = \sigma[G]$ ne contient pas de types inutiles ;*
- *$|Heads(G)| = |Heads(G')|$.*

Soit $M_{\mathcal{R}} = \max(\{\, |head^*(A_0)| \mid \exists\, R = A_1 \dots A_n \to A_0 \in \mathcal{R} \,\})$,

$$|head^*(\sigma(x))| \leq M_{\mathcal{R}} \text{ , pour tout } x \in Heads(G).$$

Démonstration. Soit $x \in Heads(G)$. Si $head(\sigma(x)) = \sigma(x)$ (c'est-à-dire si $\sigma(x)$ est primitif), alors $|head^*(\sigma(x))| = 1 \leq M_R$ donc la propriété est vérifiée.

Sinon, $head(\sigma(x)) \neq \sigma(x)$. Soit $x' \in head^*(\sigma(x))$ l'unique type tel que $pr(x') = head(\sigma(x))$ (notons que $head^*(x') = \{head(\sigma(x)), x'\}$). Par hypothèse x' est utile, donc il existe une règle $R = A_1 \dots A_n \to A_0 \in \mathcal{R}$, une substitution τ, un indice j et un type u tels que

- $\tau(A_i) \in FT(G')$ pour tout i,
- u est un sous-type de A_j,
- $\tau(u) = x'$,
- et u n'est pas primitif (car x' ne l'est pas).

Soit h la position du type consommateur dans la règle R. u est un type complexe donc u est nécessairement un sous-type de A_h. De plus par la proposition 6.5 $u \in head^*(A_h)$ car tous les arguments (indirects) de A_h sont primitifs. Or $A_0 \in head^*(A_h)$, donc par définition $A_0 \in head^*(u)$ ou $u \in head^*(A_0)$.

Montrons que $A_0 \notin head^*(pr(u))$: si $A_0 \in head^*(pr(u))$, alors $\tau(A_0) \in head^*(\tau(pr(u)))$ (proposition 6.8). Comme u n'est pas primitif $\tau(pr(u)) = pr(\tau(u))$, et $\tau(u) = x'$ donc $\tau(A_0) \in head^*(pr(x'))$. Or $x' \in head^*(\sigma(x))$ par définition, donc $\tau(A_0) \in head^*(pr(x')) \subseteq head^*(pr(\sigma(x)))$. D'après le corollaire 6.8 ceci implique que $\tau(A_0) \notin FT(G')$, ce qui constitue une contradiction. On a donc bien $A_0 \notin head^*(pr(u))$.

Par conséquent $u \in head^*(A_0)$. $\tau(A_0) \in FT(G')$, donc selon la proposition 6.11 il existe un type $t \in Lex(G')$ tel que $\tau(A_0) \in head^*(t)$. $u \in head^*(A_0)$ implique $\tau(u) \in head^*(\tau(A_0))$, ainsi $head(t) = head(\tau(A_0)) = head(\tau(u)) = head(\sigma(x))$. D'après le lemme 6.5 on peut déduire de $head(\sigma(x)) \in head^*(t)$ que $\sigma(x) \in head^*(t)$. Comme $\tau(A_0) \in head^*(t)$ on a encore $\tau(A_0) \in head^*(\sigma(x))$ ou $\sigma(x) \in head^*(\tau(A_0))$. Mais $\tau(A_0) \in FT(G)$, donc par contraposée du corollaire 6.8 $\tau(A_0) \notin head^*(pr(\sigma(x)))$, d'où $\sigma(x) \in head^*(\tau(A_0))$.

$\tau(u) = x'$ et u n'est pas un type primitif, donc par définition $\tau(pr(u)) = pr(x') = head(\sigma(x))$. $\tau(pr(u))$ étant par conséquent primitif, on a $\tau(head(u)) = \tau(pr(u)) = head(\sigma(x))$. Or $\tau(head(A_0)) = \tau(head(u))$ puisque $u \in head^*(A_0)$, d'où $\tau(head(A_0)) = head(\sigma(x))$. Ainsi $\tau(head(A_0))$ est primitif et $\sigma(x) \in head^*(\tau(A_0))$, donc par application du corollaire 6.2 $|head^*(\sigma(x))| \leq |head^*(A_0)| \leq M_{\mathcal{R}}$. □

Lemme 6.6. *Soit G une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s\rangle$ et σ une substitution fidèle telle que $\sigma[G]$ ne contient pas de types inutiles.*

Si $head(\sigma(x)) = \sigma(x)$ pour tout $x \in Heads(G)$, alors pour tout type t orphelin dans $STLex(\sigma[G])$ il existe un type t' orphelin dans $STLex(G)$ tel que $t = \sigma(t')$.

Démonstration. Soit $t \in STLex(\sigma[G])$ un type orphelin. Par le corollaire 6.3 il existe $u \in Lex(\sigma[G])$ tel que $t \in args^*(u)$, donc il existe aussi $u' \in Lex(G)$ tel que $u = \sigma(u')$. Soit $u_t \in head^*(u)$ le type tel que $t \in args(u_t)$. $args(u_t) \neq \emptyset$ et par hypothèse $\sigma(head(u'))$ est primitif, donc $u_t \notin head^*(\sigma(head(u')))$. Par conséquent il existe $t' \in args^*(u')$ tel que $\sigma(t') = t$, d'après la proposition 6.7.

Montrons maintenant que t' est orphelin : supposons qu'au contraire il existe $u'' \in Lex(G)$ tel que $t' \in head^*(u'')$. D'après la proposition 6.8 $\sigma(t') \in head^*(\sigma(u''))$, mais alors le fait que $\sigma(t') = t$ et $\sigma(u'') \in Lex(\sigma[G])$ contredit le fait que t est orphelin. □

Remarque : Étant donné qu'un type t' dans G ne peut avoir qu'une seule image par σ dans $\sigma[G]$, il découle du lemme 6.6 que $|TO(\sigma[G])| \leq |TO(G)|$.

Proposition 6.22. *Soit G une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s\rangle$ et σ une substitution fidèle telle que $\sigma[G]$ ne contient pas de types inutiles.*

Si $|Heads(G)| = |Heads(\sigma[G])|$ et $head(\sigma(x)) = \sigma(x)$ pour tout $x \in Heads(G)$ alors les deux propositions suivantes sont équivalentes :

(i) $|TO(\sigma[G])| \geq |TO(G)|$

(ii) $G \equiv \sigma[G]$.

Démonstration. $(ii) \Rightarrow (i)$. $G \equiv \sigma[G]$ si et seulement si σ est un renommage. Or si σ est un renommage alors à chaque type orphelin de G correspond un unique type orphelin de $\sigma[G]$ et réciproquement (ceci du fait que les grammaires ne contiennent pas de types inutiles, ce qui a pour conséquence par la proposition 6.14 que tous les types orphelins sont primitifs). Par conséquent $|TO(\sigma[G])| = |TO(G)|$, d'où évidemment $|TO(\sigma[G])| \geq |TO(G)|$.

$(i) \Rightarrow (ii)$. On suppose que $|TO(\sigma[G])| \geq |TO(G)|$. Montrons qu'il existe une substitution $\tau : Var(\sigma[G]) \mapsto Var(G)$ telle que $G = \tau[\sigma[G]]$.

Par hypothèse $head(\sigma(x)) = \sigma(x)$ pour tout $x \in Heads(G)$, donc pour tout $x' \in Heads(\sigma[G])$ il existe $x \in Heads(G)$ tel que $x' = \sigma(x)$: en effet, par définition s'il existe $t' \in Lex(\sigma[G])$ tel que $x' = head(t')$ alors il existe $t \in Lex(G)$ tel que $x' = head(\sigma(x))$, avec $x = head(t)$. Comme par

hypothèse $\sigma(x) = head(\sigma(x))$, on a bien $x' = \sigma(x)$. De plus $|\{x \in Heads(G) | x' = \sigma(x)\}| = 1$, car s'il existe $x \neq y \in Heads(G)$ tels que $\sigma(x) = \sigma(y) = x'$ alors $|Heads(G)| < |Heads(\sigma[G])|$. Ainsi on peut définir $\tau(x')$, pour tout $x' \in Heads(\sigma[G])$, par l'unique $x \in Heads(G)$ tel que $x' = \sigma(x)$.

Pour tout $x' \in TO(\sigma[G])$ il existe $x \in TO(G)$ tel que $\sigma(x) = x'$, d'après le lemme 6.6. Un tel x est unique, car s'il existe $x \neq y \in TO(G)$ tels que $\sigma(x) = \sigma(y) = x'$ alors $|TO(\sigma[G])| < |TO(G)|$. Ainsi il est possible de définir $\tau(x') = x$, pour tout $x' \in TO(\sigma[G])$, avec x l'unique type tel que $\sigma(x) = x'$.

Selon la proposition 6.18 $Var(\sigma[G]) \subseteq Heads(\sigma[G]) \cup TO(\sigma[G])$, donc la substitution τ est bien définie pour tout $x' \in Var(\sigma[G])$. De plus par définition $\tau = \sigma^{-1}$, donc $G = \tau[\sigma[G]]$, ce qui montre que σ est un renommage. Par conséquent $G \equiv \sigma[G]$. □

Remarque : On notera accessoirement que, sous les conditions définies pour la proposition 6.22, si x est orphelin dans G alors $\sigma(x)$ est également orphelin dans $\sigma[G]$: en effet dans le cas contraire on aurait $\sigma(x) \in head^*(\sigma(t))$ avec $t \in Lex(G)$. Comme $x \notin head^*(t)$ puisque x est orphelin il faudrait que $\sigma(x) \in head^*(\sigma(head(t))) \setminus \{head(t)\}$, ce qui implique que $\sigma(head(t))$ n'est pas un type primitif : contradiction.

Corollaire 6.9. *Soit G une grammaire sans types inutiles dans le système de $\mathcal{R}$-grammaires par consommation d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$ et σ une substitution fidèle telle que $\sigma[G]$ ne contient pas de types inutiles, $\sigma[G] \not\equiv G$ et $|Heads(\sigma[G])| = |Heads(G)|$.*

Si $|TO(\sigma[G])| \geq |TO(G)|$ alors il existe $x \in Heads(G)$ tel que $head(\sigma(x)) \neq \sigma(x)$.

Démonstration. Conséquence évidente de la proposition 6.22. □

6.4.2 Apprenabilité des grammaires par consommation stricte d'arguments

Finalement nous avons démontré que les $\mathcal{R}$-grammaires par consommation stricte d'arguments ont plusieurs spécificités favorables à l'apprenabilité, et même plus précisément à la propriété d'élasticité finie. En résumé, ces spécificités sont les suivantes :

- La relation $\sqsubseteq$ est une relation de réduction sur les $\mathcal{R}$-grammaires telle que toute classe de $\mathcal{R}$-grammaires a la densité finie bornée selon cette relation (corollaire 6.1) ;
- Tout langage d'une grammaire par consommation d'arguments est représentable à l'aide d'une grammaire ao-normale, c'est-à-dire en n'utilisant qu'un nombre limité de types orphelins (corollaire 6.7) ;
- Si $G' = \sigma[G]$, avec G et G' deux grammaires par consommation d'arguments telles que leurs nombres de types primitifs têtes sont identiques, alors soit σ diminue le nombre de types orphelins, soit σ transforme un type primitif tête en type complexe (corollaire 6.9) ;
- Enfin, la propriété clé des grammaires par consommation stricte d'arguments réside dans le fait que si $G' = \sigma[G]$, avec G et G' deux grammaires telles que leurs nombres de types primitifs têtes sont identiques, alors il existe une borne sur le nombre de sous-types têtes de $\sigma(x)$ pour tout type primitif tête x dans G (proposition 6.21).

La dernière de ces propriétés est à la base de toute la démonstration, car c'est elle qui permet de borner le nombre de grammaires obtenables par substitutions successives. Dans la suite nous montrons comment la combinaison de ces propriétés a pour conséquence l'élasticité finie des classes de grammaires concernées.

Par comparaison avec le cas des grammaires AB démontré par KANAZAWA dans [60], on constate que les conditions $|Heads(G)| = |Heads(\sigma[G])|$ et G et $\sigma[G]$ sans types inutiles n'impliquent pas que σ est une bijection. Le cas général est en effet plus complexe, et nous montrerons en revanche que le nombre de grammaires $\sigma[G]$ vérifiant ces conditions est limité, ce qui constitue une condition suffisante pour l'élasticité finie.

Proposition 6.23. *Soit $\mathcal{R}$ un ensemble de règles par consommation stricte d'arguments et k un entier.*

Si $\{G_1, G_2, \ldots, G_n\}$ est un ensemble de $\mathcal{R}$-grammaires ao-normales non vides sans types inutiles tel que

- *$|G_i| \leq k$ pour tout $i \geq 1$;*
- *$|Heads(G_i)| = |Heads(G_j)|$ pour tous $i, j \geq 1$;*
- *pour tout $i > 1$ il existe une substitution fidèle σ_i telle que $\sigma_i[G_{i-1}] = G_i$, et σ_i n'est pas un renommage,*

$$\text{alors } n \leq M_{\mathcal{R}} \times k \times (C_{ao} + 1) + 1,$$

avec $M_{\mathcal{R}} = \max(\{\ |head^*(A_0)|\ |\ \exists\ R = A_1 \ldots A_n \rightarrow A_0 \in \mathcal{R}\ \})$ et $C_{ao} = |G|^2 + 1$.

Démonstration. Soit τ_1 la substitution identité (de sorte que $G_1 = \tau_1[G_1]$). Pour tout $i > 1$, soit $\tau_i = \sigma_i \circ \sigma_{i-1} \circ \cdots \circ \sigma_2$, autrement dit τ_i est l'unique substitution telle que $G_i = \tau_i[G_1]$. τ_i est fidèle car la composition de substitutions fidèles forme toujours une substitution fidèle, par définition.

Soit $f_i(x)$ la fonction définie sur $Heads(G_1)$ par $f_i(x) = |head^*(\tau_i(x))|$ pour tout $i \geq 1$. On définit ainsi g_i et p_i de la façon suivante

$$g_i = \sum_{x \in Heads(G_1)} f_i(x),$$

$$p_i = g_i \times (C_{ao} + 1) - |TO(G_i)|.$$

Par définition $p_i \geq 0$ pour tout $i \geq 1$, car $|TO(G_i)| \leq C_{ao}$ (définition 6.12), et

- si G_i n'est pas vide alors $|Heads(G_1)| \geq 1$ d'où $g_i \geq 1$;
- sinon $|Heads(G_1)| = 0$, donc $|TO(G_1)| = |TO(G_i)| = 0$.

On montre également aisément que $f_{i-1}(x) \leq f_i(x)$ pour tout $x \in Heads(G_1)$ et tout $i > 1$: en effet, si $f_{i-1}(x) > f_i(x)$ alors $|head^*(\tau_{i-1}(x))| > |head^*(\tau_i(x))|$. Or par définition $\tau_i = \sigma_i \circ \tau_{i-1}$, et quel que soit t il est impossible que $|head^*(t)| > |head^*(\sigma_i(t))|$. Ceci a pour conséquence immédiate que $g_{i-1} \leq g_i$.

Montrons que $p_{i-1} < p_i$ pour tout $i > 1$:

- Cas 1 : $|TO(G_{i-1})| > |TO(G_i)|$. On a vu que $g_{i-1} \leq g_i$, donc il est clair dans ce cas que $p_{i-1} < p_i$: en effet $p_i - p_{i-1} = (g_i - g_{i-1}) \times C_{ao} + (|TO(G_{i-1})| - |TO(G_i)|)$, avec $(g_i - g_{i-1}) \geq 0$ et $(|TO(G_{i-1})| - |TO(G_i)|) > 0$.
- Cas 2 : $|TO(G_{i-1})| \leq |TO(G_i)|$. G_i est ao-normale donc $|TO(G_i)| \leq C_{ao}$, or $|TO(G_{i-1})| \geq 0$ donc $|TO(G_i)| - |TO(G_{i-1})| \leq C_{ao}$. $G_i = \sigma_i[G_{i-1}]$ et $|TO(G_{i-1})| \leq |TO(\sigma_i[G_{i-1}])|$, donc d'après le corollaire 6.9 il existe $x \in Heads(G_{i-1})$ tel que $\sigma_i(x) \neq head(\sigma_i(x))$, autrement dit $\sigma_i(x)$ est un type complexe. Soit $y \in Heads(G_1)$ le type tel que $x = head(\tau_{i-1}(y))$: $\sigma_i(x)$ est complexe donc $|head^*(\sigma_i(head(\tau_{i-1}(y))))| > |head^*(head(\tau_{i-1}(y)))|$, ce qui implique $|head^*(\tau_i(y))| > |head^*(\tau_{i-1}(y))|$. Par conséquent $f_{i-1}(y) < f_i(y)$, d'où $g_{i-1} < g_i$ du fait que $f_{i-1}(x) \leq f_i(x)$ pour tout x. On obtient ainsi $g_{i-1} \leq g_i - 1$:
$$\begin{aligned} g_{i-1} \times (C_{ao}+1) &\leq g_i \times (C_{ao}+1) - (C_{ao}+1) \\ &\leq g_i \times (C_{ao}+1) - (|TO(G_i)| - |TO(G_{i-1})| + 1) \\ &< g_i \times (C_{ao}+1) - |TO(G_i)| + |TO(G_{i-1})| \end{aligned}$$
Par conséquent $g_{i-1} \times (C_{ao}+1) - |TO(G_{i-1})| < g_i \times (C_{ao}+1) - |TO(G_i)|$, soit $p_{i-1} < p_i$.

$|Heads(G_n)| = |Heads(G_1)|$, G_n ne contient pas de types inutiles et il existe une substitution fidèle τ_n telle que $\tau_n[G_1] = G_n$, donc d'après la proposition 6.21, $f_n(x) \leq M_{\mathcal{R}}$ pour tout $x \in Heads(G_1)$. Par définition $|Heads(G_n)| \leq |G_n|$, or $|G_n| \leq k$, par conséquent $g_n \leq M_{\mathcal{R}} \times k$. De plus $|TO(G_n)| \geq 0$, donc $p_n = g_n \times (C_{ao}+1) - |TO(G_n)| \leq M_{\mathcal{R}} \times k \times (C_{ao}+1)$.

Ainsi on a construit une séquence d'entiers $\langle p_1, p_2, \ldots, p_n \rangle$ qui vérifie les conditions suivantes :
- $p_{i-1} < p_i$ pour tout i, $1 < i \leq n$;
- $p_1 \geq 0$;
- $p_n \leq M_{\mathcal{R}} \times k \times (C_{ao}+1)$.

Cette séquence est de taille n, donc $n \leq M_{\mathcal{R}} \times k \times (C_{ao}+1) + 1$. □

Corollaire 6.10. *Soit $\mathcal{R}$ un ensemble de règles par consommation stricte d'arguments et k un entier. Si $\{G_1, G_2, \ldots, G_n\}$ est un ensemble de $\mathcal{R}$-grammaires ao-normales sans types inutiles tel que*
- *$|G_i| \leq k$ pour tout $i \geq 1$;*
- *pour tout $i > 1$ il existe une substitution fidèle σ_i telle que $\sigma_i(G_{i-1}) = G_i$ et σ_i n'est pas un renommage,*

$$\text{alors } n \leq (M_{\mathcal{R}} \times k \times (C_{ao}+1) + 1) \times (k+1),$$

avec $M_{\mathcal{R}} = \max(\{\, |head^*(A_0)| \mid \exists\, R = A_1 \ldots A_n \to A_0 \in \mathcal{R} \,\})$.

Démonstration. Notons que si $i < j$ alors $|Heads(G_i)| \leq |Heads(G_j)|$, car il existe une substitution $\tau = \sigma_j \circ \sigma_{j-1} \circ \cdots \circ \sigma_{i+1}$ telle que $G_j = \tau[G_i]$. Soit f la fonction définie sur $\{1, \ldots, n\}$ par
- $f(1) = 1$;
- si $i > 1$, soit $E = \{\, j > f(i-1) \mid |Heads(G_{f(i-1)})| < |Heads(G_j)| \,\}$:
  - si $E \neq \emptyset$ alors $f(i) = \min(E)$;
  - sinon $f(i) = n + 1$.

Soit $m = \max(\{\, i \in \{1, \ldots, n\} \mid f(i) \leq n \,\})$, on a donc $i \leq m$ si et seulement si $f(i) \neq n+1$. Par définition de f, $|Heads(G_{f(i)})| < |Heads(G_{f(i+1)})|$ } pour tout $1 \leq i < m$. Or $0 \leq |Heads(G_{f(i)})| \leq |G_{f(i)}| \leq k$ pour tout i, donc $m \leq k+1$.

Pour tout i, $1 \leq i \leq m$, et tout j, $f(i) \leq j < f(i+1)$, on a $|Heads(G_j)| = |Heads(G_{f(i)})|$ (par définition de f). Étant donné les propriétés vérifiées par les grammaires de la séquence, la proposition 6.23 s'applique à l'ensemble de grammaires $\{G_{f(i)}, G_{f(i)+1}, \ldots, G_{f(i+1)-1}\}$, donc $f(i+1) - f(i) \leq M_{\mathcal{R}} \times k \times (C_{ao} + 1) + 1$.

Pour tout $i \geq 1$, soit s_i la séquence définie par $s_i = \langle G_{f(i)}, G_{f(i)+1}, \ldots, G_{f(i+1)-1} \rangle$. La concaténation des s_i forme la totalité de la séquence de grammaires : $s_1 \bullet s_2 \bullet \cdots \bullet s_m = \langle G_1, G_2, \ldots, G_n \rangle$. Or chaque s_i est de longueur inférieure ou égale à $M_{\mathcal{R}} \times k \times (C_{ao} + 1) + 1$ et $m \leq k+1$, donc

$$n \leq (M_{\mathcal{R}} \times k \times (C_{ao} + 1) + 1) \times (k+1).$$

□

Le théorème ci-dessous conclut la preuve de l'élasticité finie des langages de structures des $\mathcal{R}$-grammaires par consommation stricte d'arguments. Il est suivi (comme dans la plupart des cas semblables) de son corollaire pour les langages de chaînes, lorsqu'une relation finiment valuée peut être établie entre les deux types de langages.

Théorème 6.1. *Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires, avec $\mathcal{R}$ un ensemble de règles par consommation stricte d'arguments. Soit $\mathcal{G}$ une classe de grammaires sur un vocabulaire Σ dans ce système, et pour tout $k \geq 0$ soit $\mathcal{G}_k = \{\, G \in \mathcal{G} \mid |G| \leq k \,\}$.*

Pour tout $k \geq 0$, la classe de langages de structures $\{\, SL(G) \mid G \in \mathcal{G}_k \,\}$ a l'élasticité finie.

Démonstration. Soit $\mathcal{G}'$ l'ensemble des $\mathcal{R}$-grammaires ao-normales dans $\mathcal{G}_k$, et $\mathcal{G}''$ le sous-ensemble des grammaires faiblement $\sqsubseteq$-réduites dans $\mathcal{G}'$. D'après la proposition 6.1 toute grammaire faiblement réduite ne contient pas de types inutiles. Par conséquent, il découle du corollaire 6.10 qu'il existe une constante C telle que $n \leq C$ pour tout ensemble $\{G_1, G_2, \ldots, G_n\} \subseteq \mathcal{G}''$ tel qu'il existe une substitution fidèle σ_i telle que $G_i = \sigma_i[G_{i-1}]$ pour tout $i > 1$ et $G_i \not\equiv G_{i-1}$. Ceci implique selon la proposition 6.4 qu'il existe une constante C' telle que $\mathcal{G}'' \subseteq MaxRed(\mathcal{G}', \sqsubseteq, C')$.

De plus, le système de grammaires $\langle SL(\mathcal{R})_\Sigma, \mathcal{G}, SL \rangle$ a la densité finie bornée selon $\sqsubseteq$, d'après le corollaire 6.1. Par application de la proposition 5.4, on obtient que la classe $\{\, SL(G) \mid G \in \mathcal{G}' \,\}$ a l'élasticité finie. Or d'après le corollaire 6.7 pour toute grammaire G de $\mathcal{G}_k$ il existe une grammaire ao-normale G' dans $\mathcal{G}'$ telle que $SL(G) = SL(G')$ et $|G'| \leq |G| \leq k$, c'est-à-dire $G' \in \mathcal{G}'$. On obtient donc

$$\{\, SL(G) \mid G \in \mathcal{G}' \,\} = \{\, SL(G) \mid G \in \mathcal{G}_k \,\},$$

par conséquent la classe $\{\, SL(G) \mid G \in \mathcal{G}_k \,\}$ a également l'élasticité finie. □

Rappelons que dans le cas des $\mathcal{R}$-grammaires algébriques une règle $A_1 \dots A_n \rightarrow A_0$ est strictement décroissante si $n \geq 2$ (définition 4.14), autrement dit la partie gauche de la règle contient au moins deux types.

Corollaire 6.11. *Soit $\langle \mathcal{O}, \mathcal{R}, s \rangle$ un système de $\mathcal{R}$-grammaires, avec $\mathcal{R}$ un ensemble de règles strictement décroissantes par consommation stricte d'arguments. Soit $\mathcal{G}$ une classe de grammaires sur un vocabulaire Σ dans ce système, et pour tout $k \geq 0$ soit $\mathcal{G}_k = \{ G \in \mathcal{G} \mid |G| \leq k \}$.*

Pour tout $k \geq 0$, la classe de langages de chaînes $\{ L(G) \mid G \in \mathcal{G}_k \}$ a l'élasticité finie.

Démonstration. Rappelons que $L(G) = \{ prod(S) \mid S \in SL(G) \}$ (définition 4.8). Soit R la relation définie sur $\Sigma^* \times SL(\mathcal{R})$ par $x \ R \ S$ si et seulement si $x \in prod(S)$ (autrement dit x est la phrase représentée par la structure S) : on remarque que pour toute $\mathcal{R}$-grammaire G $R^{-1}[SL(G)] = \{ x \in \Sigma^* \mid \exists S \in SL(G)$ tel que $x \in prod(S) \} = L(G)$. Soit $G \in \mathcal{G}_k$ une $\mathcal{R}$-grammaire k-bornée. $\mathcal{R}$ est un ensemble de règles strictement décroissantes, donc pour tout x appartenant à $L(G)$ le nombre de $\mathcal{R}$-structures telles que $x \in prod(S)$ est borné : en effet, si $|x| = n$, une telle structure contient n nœuds de règles locales et au maximum $n - 1$ nœuds de règles universelles. La relation R est donc finiment valuée, or d'après la proposition 5.1 pour tout $k \geq 0$ la classe $\{ SL(G) \mid G \in \mathcal{G}_k \}$ a l'élasticité finie. Par conséquent, d'après la proposition 2.14, la classe $\{ L(G) \mid G \in \mathcal{G}_k \}$ a aussi l'élasticité finie. □

Concernant les langages de chaînes, KANAZAWA avait également démontré que pour toute grammaire AB G contenant n règles il existe une grammaire de constituants algébrique G' d'au plus $2n - 1$ règles telle que $L(G) = L(G')$. Ce résultat est intéressant dans la mesure où il permet de démontrer de façon différente l'élasticité finie des langages de chaînes des grammaires AB k-valuées : en effet, la classe des grammaires AB k-valuées est ainsi entièrement incluse dans la classe des langages de grammaires algébriques d'au plus $2k|\Sigma| - 1$ règles. Or cette classe a l'élasticité finie d'après le théorème de SHINOHARA (corollaire 2.4, p. 67). Il est donc pertinent d'étudier la possibilité de borner le nombre de règle d'une grammaire algébrique équivalente à une grammaire par consommation stricte d'arguments : si cela était possible, alors la classe des langages de chaînes des grammaires par consommation stricte d'arguments serait apprenable, et non seulement la sous-classe des grammaires utilisant des règles strictement décroissantes.

Malheureusement une telle équivalence n'existe pas dans le cas des $\mathcal{R}$-grammaires par consommation stricte d'arguments. Pour le montrer nous proposons ci-dessous un point limite (théorème 2.1) pour les langages de chaînes des $\mathcal{R}$-grammaires par consommation stricte d'arguments *rigides* : l'existence de celui-ci implique que cette classe n'est pas apprenable (donc a fortiori celle des grammaires k-valuées), et par conséquent n'est pas incluse dans la classe des grammaires algébriques d'au plus k règles, quel que soit k.

Exemple 6.2.

$$\text{Soit } \mathcal{R} = \left\{ \begin{array}{lll} (R_{/_1^?}) & A/^?B \quad B \rightarrow A & (\text{avec } Var(R_{/_1^?}) = \{A, B\}) \ ; \\ (R_{/_\varepsilon^?}) & A/^?B \rightarrow A & (\text{avec } Var(R_{/_\varepsilon^?}) = \{A, B\}) \end{array} \right\}$$

Notons que $\mathcal{R}$ est un ensemble de règles par consommation stricte d'arguments.

La classe des $\mathcal{R}$-grammaires admet le point limite suivant :

Soit $G_n = \{\ x \triangleright a\ ;\ b \triangleright A_n\ \}$, avec $A_0 = s$ et $X_i = X_{i-1}/^? a$ pour tout $i > 0$.

On constate aisément que $L(G_n) \subset L(G_{n+1})$ pour tout $n \geq 0$, puisque $L(G_n) = \{\ ba^i \mid i \leq n\}$. Soit $G_\infty = \{\ x \triangleright a/^? a\ ;\ b \triangleright s/^? a\ \} : L(G_\infty) = \{ba^*\}$, donc $L(G_\infty)$ constitue un point limite pour cette classe de $\mathcal{R}$-grammaires.

L'exemple ci-dessus montre que la classe des $\mathcal{R}$-grammaires par consommation stricte d'arguments n'est pas apprenable à partir de chaînes.

6.5 Applications et résultats connexes

6.5.1 Classes de $\mathcal{R}$-grammaires par consommation stricte d'arguments

Les critères nécessaires à l'application de ce résultat d'apprenabilité n'imposent pas de contraintes "externes" sur la forme des types dans les grammaires (du type borne sur la taille), comme c'était le cas avec le résultat présenté précédemment dans le chapitre 5. En revanche, ces critères sont extrêmement stricts sur la forme des règles universelles autorisées : les opérateurs sont obligatoirement orientés (définition 6.4), et les règles par consommation stricte d'arguments doivent remplir un ensemble de conditions très contraignantes (définition 6.7). Ces restrictions affaiblissent malheureusement beaucoup l'intérêt d'un tel résultat, parce qu'elles excluent de fait la plupart des formalismes grammaticaux intéressants.

Bien entendu, dans la mesure où ce résultat est une généralisation de celui de KANAZAWA pour les grammaires AB, on retrouve tout de même ces dernières parmi les systèmes auxquels le résultat s'applique :

$$\mathcal{R}_{AB} = \left\{ \begin{array}{lll} (R_{FA}) & A/B \quad B \rightarrow A & (\text{avec } Var(R_{FA}) = \{A, B\})\ ; \\ (R_{BA}) & B \quad B\backslash A \rightarrow A & (\text{avec } Var(R_{BA}) = \{A, B\}) \end{array} \right\}$$

En définissant de façon classique les positions arguments des opérateurs par $arg_/(1) = 0$, $arg_/(2) = 1$, $arg_\backslash(1) = 1$ et $arg_\backslash(2) = 0$, on constate qu'il s'agit bien d'opérateurs orientés. De plus les règles FA et BA vérifient bien l'ensemble des conditions des règles par consommation stricte d'arguments : $A \in head^*(A/B)$, $args^*(A/B) = \{B\} \subseteq Var(R_{FA})$ et $B \in args^*(A/B)$ (on vérifie facilement les mêmes conditions pour la règle BA de façon symétrique).

Mais parmi les autres formalismes connus susceptibles d'être intégrés à ce résultat, aucun ne satisfait les critères requis : même les règles des grammaires catégorielles combinatoires de STEEDMAN sont trop complexes. Finalement, les grammaires catégorielles avec itérations (présentées dans la partie 4.3.2) semblent être le seul formalisme pour lequel il existe un sous-ensemble de règles obéissant à ces critères. Plus exactement, les seuls opérateurs utilisables dans contexte sont les opérateurs $/^?$ et $\backslash^*$, dont les règles sont :

$$\mathcal{R}_? = \left\{ \begin{array}{lll} (R_{/^?_1}) & A/^?B \quad B \to A & (\text{avec } Var(R_{/^?_1}) = \{A, B\}) ; \\ (R_{/^?_\varepsilon}) & A/^?B \to A & (\text{avec } Var(R_{/^?_\varepsilon}) = \{A, B\}) ; \\ (R_{\backslash^?_1}) & B \quad B\backslash^?A \to A & (\text{avec } Var(R_{\backslash^?_1}) = \{A, B\}) ; \\ (R_{\backslash^?_\varepsilon}) & B\backslash^?A \to A & (\text{avec } Var(R_{\backslash^?_\varepsilon}) = \{A, B\}) ; \end{array} \right\}$$

Avec les définitions habituelles $arg_{/^?}(1) = 0$, $arg_{/^?}(2) = 1$, $arg_{\backslash^?}(1) = 1$ et $arg_{\backslash^?}(2) = 0$, ces règles vérifient elles aussi toutes les conditions des règles par consommation stricte d'arguments. On notera notamment que rien n'interdit qu'il n'y ait qu'une seule occurence de B dans les règles $R_{/^?_\varepsilon}$ et $R_{\backslash^?_\varepsilon}$. On peut donc conclure à cette modeste conséquence du théorème 6.1 : la classe des langages de $\mathcal{R}$-structures des familles de grammaires k-valuées (pour tout $k \in \mathbb{N}$) utilisant tout ou partie des règles $\mathcal{R}_{AB} \cup \mathcal{R}_?$ sont apprenables. On ne peut malheureusement pas en dire autant des langages de chaînes, puisque les règles $R_{/^?_\varepsilon}$ et $R_{\backslash^?_\varepsilon}$ ne sont pas à diminution strictement limitée. Le corollaire 6.11 s'applique toutefois aux règles FA et BA, on retrouve alors le résultat de KANAZAWA sur l'élasticité finie des langages de chaînes des grammaires AB k-valuées.

Il est clair que le résultat que nous avons présenté est assez pauvre en termes d'applications. Il faut tout de même signaler qu'il est possible que d'autres règles que celles présentées ci-dessus remplissent les critères requis, même si peu de variations sont possibles. Par exemple, une règle de la forme

$$A \quad B \quad f(A, g(D, g(C, E)), B, D) \quad D \to g(C, E)$$

est correcte, avec $arg_f(1) = arg_f(3) = arg_f(4) = 1$, $arg_f(2) = 0$, $arg_g(1) = 1$ et $arg_g(2) = 0$. En effet on a bien :

- $g(C, E) \in head^*(f(A, g(D, g(C, E)), B, D))$;
- $args^*(f(A, g(D, g(C, E)), B, D)) = \{A, B, C, D\} \subseteq Var(R)$;
- $\{A, B, D\} \subseteq args^*(f(A, g(D, g(C, E)), B, D))$;
- Il n'y a qu'une seule occurence de C dans $f(A, g(D, g(C, E)), B, D)$.

Évidemment, on voit mal l'intérêt que pourrait avoir ce type de règle sur le plan linguistique. Cet exemple n'a d'autre but que d'illustrer la possible inclusion dans notre résultat de règles dont la forme est très éloignée de celle des grammaires AB. Même si cela demeure très hypothétique, on peut envisager que cela soit utile à certains formalismes grammaticaux non traités ici.

6.5.2 Élasticité infinie des $\mathcal{R}$-grammaires par consommation d'arguments

Le résultat positif que nous avons montré concernant l'élasticité finie des grammaires par consommation stricte d'arguments donne une réponse partielle à la question ouverte de KANAZAWA. Mais ce résultat est peu significatif seul, d'une part parce qu'il est très limité et d'autre part parce qu'il ne permet pas de situer de façon plus précise la frontière entre les classes de langages ayant l'élasticité finie et les autres. Nous proposons donc ci-dessous un résultat négatif d'élasticité finie pour les grammaires par consommation d'arguments en général.

6.5.2.1 Contre-exemple à l'élasticité finie

Exemple 6.3. Soit $\mathcal{R}$ l'ensemble de règles universelles défini par les règles

$$\begin{array}{ll} / & A/B\ B \to A \\ /_i^+ & A/^+B\ B \to A/^+B \\ /_\varepsilon^+ & A/^+B\ B \to A \end{array}$$

Soit $G_0 = \{\ x : s/a_0\ ;\ y : s/b_0\ ;\ z : a_0\ ;\ w : b_0\ \}$, et pour tout $i > 0$ soit σ_i la substitution définie par :

$$\sigma_i = \begin{cases} \{\ a_{i-1} \mapsto a_i/^+b_i\ ;\ b_{i-1} \mapsto b_i\ \} \text{ si } i \text{ est impair ;} \\ \{\ a_{i-1} \mapsto a_i\ ;\ b_{i-1} \mapsto b_i/^+a_i\ \} \text{ si } i \text{ est pair.} \end{cases}$$

Pour tout $i > 0$, soit $G_i = \sigma_i[G_{i-1}]$.

Le fait que $SL(G_{i-1}) \subseteq SL(G_i)$ (pour tout $i > 0$) est une conséquence directe de la proposition 4.11. Nous allons montrer qu'il existe parallèlement à la séquence $\langle G_i \rangle_{i \geq 0}$ une séquence $\langle e_i \rangle_{i>0}$ telle que $e_i \notin G_j$ pour tout $j < i$.

Proposition 6.24. *Pour tout $i > 0$, il existe un arbre de dérivation complet pour la grammaire G_i contenant*

- *si i est pair, un sous-arbre de racine a_i et un sous-arbre de racine $b_i/^+a_i$;*
- *si i est impair, un sous-arbre de racine $a_i/^+b_i$ et un sous-arbre de racine b_i.*

Démonstration. Par induction sur i.

$i = 1$. $G_1 = \{\ x : s/(a_1/^+b_1)\ ;\ y : s/b_1\ ;\ z : a_1/^+b_1\ ;\ w : b_1\ \}$, et l'arbre

$$/(s, l(s/(a_1/^+b_1), x), /_i^+(a_1/^+b_1, l(a_1/^+b_1, z), l(b_1, w)))$$

contient bien un nœud de type $a_1/^+b_1$ et un nœud de type b_1.

$i > 1$. Cas 1 : i est impair. On suppose que la propriété est vraie pour $i-1$: il existe un arbre de dérivation pour G_{i-1} contenant un sous-arbre de racine a_{i-1} et un sous-arbre de racine $b_{i-1}/^+a_{i-1}$ (car $i-1$ est pair). $G_i = \sigma_i[G_{i-1}]$, avec $\sigma_i = \{\ a_{i-1} \mapsto a_i/^+b_i\ ;\ b_{i-1} \mapsto b_i\ \}$, donc il existe un arbre de dérivation T pour G_i contenant

- un sous-arbre T' de racine $\sigma_i(a_{i-1}) = a_i/^+b_i$
- et un sous-arbre T'' de racine $\sigma_i(b_{i-1}/^+a_{i-1}) = b_i/^+(a_i/^+b_i)$.

Soit $T''' = /_\varepsilon^+(b_i, T'', T')$: cet arbre partiel est correctement formé car $b_i/^+(a_i/^+b_i)\ \ a_i/^+b_i \to_{/_\varepsilon^+} b_i$, et les arbres T' et T'' existent. En insérant l'arbre $/_i^+(a_i/^+b_i, T', T''')$ à l'emplacement de T' dans T, on obtient un arbre de dérivation complet pour G_i contenant les sous-arbres T' de racine $a_i/^+b_i$ et T''' de racine b_i.

Cas 2 : i est pair. On suppose que la propriété est vraie pour $i-1$: il existe un arbre de dérivation pour G_{i-1} contenant un sous-arbre de racine $a_{i-1}/^+b_{i-1}$ et un sous-arbre de racine b_{i-1} (car $i-1$ est impair). $G_i = \sigma_i[G_{i-1}]$, avec $\sigma_i = \{\ a_{i-1} \mapsto a_i\ ;\ b_{i-1} \mapsto b_i/^+a_i\ \}$, donc il existe un arbre de dérivation T pour G_i contenant

- un sous-arbre T' de racine $\sigma_i(a_{i-1}/^+b_{i-1}) = a_i/^+(b_i/^+a_i)$
- et un sous-arbre T'' de racine $\sigma_i(b_{i-1}) = b_i/^+a_i$.

Soit $T''' = /^+_\varepsilon(a_i, T', T'')$: cet arbre partiel est correctement formé car $a_i/^+(b_i/^+a_i)\ \ b_i/^+a_i \rightarrow_{/^+_\varepsilon} a_i$, et les arbres T' et T'' existent. En insérant l'arbre $/^+_i(b_i/^+a_i, T'', T''')$ à l'emplacement de T'' dans T, on obtient un arbre de dérivation complet pour G_i contenant les sous-arbres T'' de racine $b_i/^+a_i$ et T''' de racine a_i. □

Proposition 6.25. *Il existe une séquence de $\mathcal{R}$-grammaires $\langle G'_i\rangle_{i\geq 0}$ et une séquence de $\mathcal{R}$-structures $\langle e_i\rangle_{i>0}$ telle que $e_i \notin G'_j$ pour tout $j < i$.*

Démonstration. On définit la séquence infinie $\langle G'_i\rangle_{i\geq 0}$ comme la sous-séquence des G_i définies dans l'exemple 6.3 formée uniquement des indices pairs : soit $G'_i = G_{2i}$ pour tout $i \geq 0$.

Selon la proposition 6.24, pour tout $i > 0$ il existe un arbre de dérivation complet pour G_{2i} contenant un sous-arbre de racine a_{2i} : soit e_i la $\mathcal{R}$-structure dont cet arbre est l'instance, $e_i \in SL(G'_i)$. Clairement la forme de toute grammaire G'_i impose que tout type a_i soit formé à partir de la règle $z \mapsto A_i$, avec $A_i = (\tau_i \circ \tau_{i-1} \circ \cdots \circ \tau_1)(a_0)$ (avec $\tau_{2i} = \sigma_i \circ \sigma i - 1$), à l'aide d'un sous-arbre dont la branche principale passe par tous les éléments de $head^*(A_i)$. Ceci ne peut se faire qu'en utilisant $|head^*(A_i)|$ fois la règle $/^+_\varepsilon$. Mais $|head^*(A_i)| = i$, donc cela n'est possible que pour les grammaires G'_j telles que $j \geq i$. □

Théorème 6.2. *Il existe des classes de $\mathcal{R}$-grammaires par consommation d'arguments dont les classes de langages correspondantes ont l'élasticité infinie.*

Démonstration. Nous avons exhibé une telle classe dans l'exemple 6.3, dont nous avons prouvé l'élasticité infinie dans la proposition 6.25. □

Il convient de rappeler que l'élasticité infinie d'une classe de langages n'implique pas la non apprenabilité de cette classe. Ce contre-exemple est seulement destiné à montrer que la méthode utilisée sur les grammaires AB par KANAZAWA dans [60], qui repose sur l'élasticité finie, n'est pas généralisable à l'ensemble des grammaires par consommation d'arguments. La question de l'existence de classes de langages (de structures) non apprenables parmi les grammaires par consomation d'arguments est laissée ouverte.

6.5.2.2 Discussion

il est intéressant de constater que l'ensemble $\mathcal{R}$ proposé dans l'exemple 6.3 a certaines propriétés dont on aurait pu supposer qu'elles permettraient au contraire d'aboutir à l'élasticité finie. En effet on peut voir que toutes les règles respectent la condition de "consommation obligatoire d'arguments" :

Définition 6.15 (Règles par consommation obligatoire d'arguments)**.** Soit $R = A_1 \dots A_n \rightarrow B$ une règle par consommation d'arguments, dont A_h est le type consommateur. R est une règle *par consommation obligatoire d'arguments* si l'équivalence suivante est vérifiée :

$1 \leq i \leq n$ et $i \neq h$ si et seulement si $A_i \in args^*(A_h)$.

En d'autres termes, de telles règles imposent que tout argument du type consommateur soit effectivement consommé (alors que dans le cas général, les arguments peuvent "disparaître" du type consommateur sans qu'un type identique à cet argument ne soit présent dans la dérivation). Cette propriété présente l'avantage de garantir la présence d'un sous-arbre strict pour chaque argument d'un type t dans un arbre de dérivation de racine t, comme le montre la proposition ci-dessous.

Proposition 6.26. *Soit $G = \langle \Sigma, Pr, \triangleright \rangle$ une grammaire dans le système de $\mathcal{R}$-grammaires par consommation obligatoire d'arguments $\langle \mathcal{O}, \mathcal{R}, s \rangle$.*

S'il existe un arbre de dérivation (partiel) P tel que $racine(P) = t$, alors il existe $t' \in Lex(G)$ et $w \in prod(P)$ tels que les conditions suivantes sont vraies :

(i) $w \triangleright_G t'$;

(ii) $t \in head^*(t')$;

(iii) pour tout $u \in head^*(t') \setminus head^*(t)$ et tout $u' \in args(u)$, P contient strictement un sous-arbre dont la racine est u'.

Démonstration. Les conditions (i) et (ii) sont démontrées dans la proposition 6.11, nous donnons donc ici seulement la partie de la preuve spécifique à la condition (iii). Par induction sur la hauteur de l'arbre $h(P)$.

- $h(P) = 0$. dans ce cas $t = t'$ donc $head^*(t') \setminus head^*(t)$ est vide : évident.
- $h(P) > 0$. Il existe $t' \in Lex(G)$ tel que $w \triangleright t'$ et $t \in head^*(t')$ (proposition 6.11). Soit $P = [R](t, P_1, \ldots, P_n)$ et $t_i = racine(P_i)$ pour tout i. Soit h la position du type consommateur dans la règle R, t_h étant le type consommateur. $t \in head^*(t_h)$, donc par définition $head^*(t') \setminus head^*(t) = (head^*(t') \setminus head^*(t_h)) \cup (head^*(t_h) \setminus head^*(t))$. Soit $u \in head^*(t') \setminus head^*(t)$:

  1. si $u \in head^*(t') \setminus head^*(t_h)$: par hypothèse d'induction la propriété est vraie sur P_h car $h(P_h) < h(P)$. De plus tout sous-arbre strict de P_h est aussi un sous-arbre strict de P, donc la propriété (iii) est vérifiée.
  2. si $u \in (head^*(t_h) \setminus head^*(t))$: soit $R = A_1 \ldots A_n \rightarrow A_0$ et σ la substitution telle que $\sigma(A_i) = t_i$ pour tout $i > 0$ et $\sigma(A_0) = t$. On a donc $u \in (head^*(\sigma(A_h)) \setminus head^*(\sigma(A_0)))$. Ceci implique qu'il existe $a \in head^*(A_h) \setminus head^*(A_0)$ tel que $\sigma(a) = u$ (car $head^*(\sigma(A_h)) \setminus head^*(\sigma(A_0)) = \{ \sigma(a) \mid a \in head^*(A_h) \setminus head^*(A_0) \}$). $a \notin head^*(A_0)$ donc $a \neq head(A_h)$, par conséquent a est un type complexe. Comme $\sigma(a) = u$ et $u' \in args(u)$, il existe $a' \in args(a)$ tel que $\sigma(a') = u'$. Or $a' \in args^*(A_h)$ et R est une règle par consommation obligatoire d'arguments, donc il existe $i \neq h$ tel que $A_i = a'$, d'où $t_i = \sigma(A_i) = \sigma(a') = u'$. t_i est la racine du sous-arbre strict P_i dans P, par conséquent la propriété (iii) est vérifiée.

□

Cette propriété rejoint partiellement la propriété décrite dans la proposition 6.20 pour les grammaires par consommation stricte d'arguments, dans le sens où ici la construction d'un type nécessite auparavant la construction de tous ses arguments. Mais contrairement au cas des grammaires par consommation

stricte d'arguments (corollaire 6.8), on voit ici dans l'exemple 6.3 qu'un sous-type strict de $\sigma(x)$ (avec $x \in Heads(G)$) peut très bien être un type faisable pour la grammaire $\sigma[G]$. Ainsi il n'y a pas de limite dans les combinaisons de types "nouveaux" (issus d'une substitution) et "anciens" (images de types existants avant la substitution), donc pas de limite au nombre de grammaires différentes qu'on peut obtenir par substitution.

Ces considérations peuvent sembler être des détails techniques peu pertinents, pourtant cet aspect mérite un tel éclaircissement car le lemme clé (*Key lemma*) 5.56 (p. 76 dans [60], reprise dans ce document en tant que proposition 3.10, p. 79) de KANAZAWA est relativement ambigu sur ce point. En effet, la démonstration proposée par KANAZAWA utilise la notion de "dépendance" d'un type sur un autre (*x depends on y*) : un type primitif a dépend d'un type primitif b si tous les types de $Lex(G)$ ayant a pour type tête ont b pour argument, c'est-à-dire si $t \in Lex(G)$ et $a = head(t)$ alors $b \in args^*(t)$. La démonstration se conclut en montrant que sous les contraintes que $|Heads(G)| = |Heads(\sigma[G])|$, $|Var(G)| = |Var(\sigma[G])|$, avec G et G' sans types inutiles, si σ n'est pas un renommage alors il existe une chaîne infinie de dépendances entre types primitifs. Cette chaîne infinie implique qu'il existe un type dépendant de lui-même, ce type est donc inutile d'où contradiction : σ ne peut être qu'un renommage. Le

léger flou sur la notion de "type dépendant d'un autre" n'empêche pas la démonstration d'être valide pour le cas des grammaires AB, car effectivement un type qui dépend de lui-même dans ce cadre est inutile. En fait cela est dû au fait que les règles des grammaires AB sont à la fois des règles par consommation obligatoire d'arguments et par consommation stricte d'arguments : la notion de dépendance utilisée semble insister sur le fait qu'un type b est nécessaire pour construire un type a (consommation obligatoire), alors que c'est le fait que l'argument soit toujours éliminé du type consommateur (consommation stricte) qui joue le rôle le plus important pour parvenir à l'élasticité finie.

6.6 Conclusion

Dans [60], la question de l'existence de classes de grammaires combinatoires générales (autres que les grammaires AB) ayant l'élasticité finie était laissée ouverte. Nous avons donné dans ce chapitre une condition suffisante sur les ensembles de règles universelles pour que les classes de $\mathcal{R}$-grammaires correspondantes k-bornées aient l'élasticité finie. Comme on l'a vu, cette condition est extrêmement restrictive, et ne s'éloigne de fait que très peu du cas des grammaires AB, pour lesquelles l'élasticité finie a été démontrée par KANAZAWA dans [60]. Mais nous avons également démontré qu'en s'éloignant seulement "un peu plus" du cas où la condition suffisante que nous proposons est remplie, on obtenait au contraire l'élasticité infinie. Bien entendu rien ne prouve qu'il n'y ait pas des classes de GCG avec l'élasticité finie "ailleurs", c'est-à-dire en ne recherchant plus cette propriété selon la méthode employée pour les grammaires AB. Cependant les grammaires AB étant l'unique cas connu à ce jour de classe de langage suffisamment riche avec l'élasticité finie, on est en droit de supposer que nos résultats signifient que la frontière séparant les classes de grammaires combinatoires générales ayant l'élasticité finie de celles qui ne l'ont pas est finalement très proche des grammaires AB.

COSTA FLORÊNCIO avait étudié cette même question dans [32] (ainsi que dans sa thèse [35]). Dans

ce travail, COSTA FLORÊNCIO propose une transformation générique de règles universelles en un ensemble de règles de forme identique à celle des grammaires AB, de manière à rapporter le problème de l'élasticité finie des classes de grammaires utilisant des règles transformables de cette façon au cas déjà démontré par KANAZAWA des grammaires AB. À l'aide du théorème de KANAZAWA concernant l'élasticité finie avec une relation finiement valuée (théorème 2.14), il obtient une condition suffisante sur les règles universelles pour l'élasticité finie des classes de grammaires ainsi transformée. Cependant ce résultat comporte un inconvénient non négligeable : la transformation requise pour convertir les grammaires de sorte que celles-ci utilisent les règles équivalentes similaires aux grammaires AB ne préserve pas la borne sur le nombre de règles. Ceci signifie que pour une classe de $\mathcal{R}$-grammaires $\mathcal{G}$ k-bornée donnée, l'existence d'un k' tel que l'image de $\mathcal{G}$ par la transformation soit k'-bornée n'est pas garantie, car le nombre de règle d'une grammaire transformée dépend notamment du degré des types de la grammaire d'origine :

> *La valeur de k dépend à la fois de R et (du degré) des types de la grammaire G (qui est fini mais non borné). [35]*

C'est pourquoi le résultat de COSTA FLORÊNCIO ne peut s'interpréter que dans le contexte de la transformation qu'il propose, ce qui en limite la portée. À l'inverse, notre résultat donne des conditions suffisantes (qui portent aussi sur les règles) aboutissant directement à l'élasticité finie de la classe de grammaires k-bornées correspondante.

Finalement, nous avons montré que le critère des grammaires par consommation stricte d'arguments est une condition suffisante pour l'élasticité finie des grammaires combinatoires générales (lexicalisées et algébriques) k-bornées. Ceci a pour conséquences l'apprenabilité des classes de langages de $\mathcal{R}$-structures de ces grammaires, et celle des langages de chaînes sous la restriction des règles strictement décroissantes. Le critère utilisé impose des contraintes fortes sur la forme des règles universelles. Mais contrairement à la condition suffisante des grammaires à arguments bornés présentée dans le chapitre 5, il ne requiert pas de limite particulière sur les types utilisés dans les grammaires. Il s'agit donc d'un résultat plus clair, au sens où la classe de langages complète est apprenable pour les familles de grammaires concernées (toujours sous la restriction du nombre de règles borné bien sûr). Nous avons ainsi obtenu deux résultats d'apprenabilité complémentaires, l'un relativement riche sur le plan des formalismes grammaticaux qui y sont potentiellement inclus mais limité au niveau de la forme des grammaires, l'autre plus restreint sur les formalismes mais non limité par la forme des grammaires.

PARTIE III

Vers des applications ?

CHAPITRE 7

Apprentissage partiel et application aux grammaires de liens

D'UN POINT DE VUE THÉORIQUE, NOUS AVONS VU DANS LES CHAPITRE PRÉCÉDENTS QU'UN CERTAIN NOMBRE DE CLASSES DE LANGAGES SONT APPRENABLES DANS LE MODÈLE DE GOLD. PARMI CELLES-CI, NOUS AVONS MONTRÉ QUE CERTAINES PERMETTENT DE REPRÉSENTER DE FAÇON ASSEZ SATISFAISANTE LES LANGUES NATURELLES, À L'AIDE DE FORMALISMES GRAMMATICAUX RELATIVEMENT BIEN ADAPTÉS. MAIS LA MISE EN ŒUVRE CONCRÈTE DE L'APPRENTISSAGE AUTOMATIQUE SOULÈVE DE NOMBREUSES DIFFICULTÉS, EN TERMES D'EFFICACITÉ ALGORITHMIQUE ET DE DISPONIBILITÉ DES DONNÉES NÉCESSAIRES À CELLE-CI NOTAMMENT. LA MAJORITÉ DES ALGORITHMES D'APPRENTISSAGE EXISTANTS UTILES (PERMETTANT D'ACQUÉRIR DES CLASSES DE LANGAGES INTÉRESSANTES) SONT EN EFFET DE COMPLEXITÉ EXPONENTIELLE. NOUS ABORDONS DANS CE CHAPITRE UN ASPECT DE CE PROBLÈME, CELUI DU MANQUE D'EFFICACITÉ DANS LE CAS DE L'APPRENTISSAGE À PARTIR DE SIMPLES PHRASES. NOUS PROPOSONS DE RENDRE LE CONTEXTE DE L'APPRENTISSAGE PLUS SOUPLE, À TRAVERS LE PRINCIPE DE L'APPRENTISSAGE PARTIEL. NOUS MONTRONS EN QUOI CETTE MÉTHODE EST PERTINENTE, À L'AIDE D'UNE EXPÉRIMENTATION AVEC LES GRAMMAIRES DE LIENS, PORTANT SUR DES DONNÉES DE TAILLE MOYENNE.

7.1 Introduction

Il est clair que le critère d'identification à la limite, défini comme base de l'apprenabilité, fait du modèle de GOLD une formalisation uniquement théorique du processus d'apprentissage. En effet, si ce critère garantit bien l'existence d'un algorithme d'apprentissage pour les classes de langages apprenables (autrement dit la décidabilité du problème de l'identification à la limite pour cette classe de langages), il ne donne aucune assurance d'une quelconque notion d'efficacité pour cet algorithme. C'est pourquoi la plupart des algorithmes d'apprentissage universels sont totalement inutilisables en pratique : typiquement, il n'est en aucun cas envisageable d'énumérer par ordre lexicographique l'ensemble des grammaires de la classe étudiée après chaque nouvel exemple. Ainsi, la mise en œuvre de l'apprentissage nécessite au minimum que l'algorithme soit capable de construire une grammaire à partir des

données dont il dispose, sans avoir à effectuer une recherche systématique dans l'ensemble de la classe de grammaires. De plus, il est souhaitable que la construction de la grammaire soit réalisable en un temps raisonnable.

De tels algorithmes d'apprentissage existent, bien sûr : c'est notamment le cas de l'algorithme RG proposé initialement par BUSZKOWSKI, qui apprend la classe des langages de FA-structures des grammaires AB rigides (voir partie 3.2). Cet algorithme est de complexité polynomiale, de même que la généralisation (présentée dans la partie 4.4.2) qui en est tirée pour les grammaires combinatoires générales[1]. Mais par rapport aux applications aux langues naturelles, ce type d'algorithme présente deux défauts majeurs (détaillés dans la partie 3.2.4) : la limitation aux grammaires rigides et la nécessité de structures complexes en entrée. Pour les grammaires AB, KANAZAWA a proposé un algorithme qui apprend la classe des langages de chaînes de grammaires AB k-valuées, mais celui-ci est de complexité exponentielle. COSTA FLORÊNCIO a démontré dans [34] et [33] qu'il s'agit en fait d'un problème NP-dur (voir partie 3.3).

Dans ce chapitre nous abordons le problème de la mise en œuvre d'algorithmes d'apprentissage sur des données réelles en langage naturel. Ce problème étant très complexe, nous nous limitons[2] ici à étudier la question de la nature des données requises par l'algorithme en entrée. Dans le cadre des algorithmes de type RG, il s'agit de trouver le moyen d'apprendre "en temps raisonnable", sans pour autant avoir besoin de structures complexes telles que les FA-structures. Autrement dit, nous cherchons à apprendre un langage à partir de phrases simples, en limitant l'explosion combinatoire due à la multiplicité des structures possibles pour une phrase. Dans ce but, nous proposons d'assouplir le cadre de l'apprentissage, en fournissant à l'algorithme une partie de la grammaire qu'il doit apprendre. Nous présentons donc un algorithme d'*apprentissage partiel*, dont le fonctionnement doit permettre de tirer profit des informations de la *grammaire initiale* pour pallier l'absence des informations qui auraient été apportées par des structures.

Nous évaluons ensuite la faisabilité de l'approche proposée, à l'aide du lexique anglais proposé par les auteurs des grammaires de liens SLEATOR et TEMPERLEY. Même si ces données sont relativement simples et de taille limitée, il s'agit d'une grammaire assez complète de l'anglais. En ce sens, cette première expérimentation est représentative des problèmes que pose le passage à l'échelle de véritables données textuelles pour l'apprentissage automatique. Les résultats obtenus semblent indiquer que l'apprentissage partiel répond de façon satisfaisante au problème de la nature des données requises par l'algorithme d'apprentissage.

L'acquisition automatique de grammaires d'après corpus est une problématique qui a déjà fait l'objet de plusieurs travaux. Dans le domaine des grammaires catégorielles, HOCKENMAIER a réalisé l'acquisition d'un lexique de l'anglais sous forme de grammaires catégorielles combinatoires, à partir du *Penn Treebank* [55], [30]. MOOT a réalisé l'acquisition d'un lexique de grammaires catégorielles multimodales pour son prouveur de théorèmes *Grail* [71], [73], [72]. Il est clair que ces précédents travaux sont

[1] Sous réserve que la relation $\sim$, qui est un paramètre de l'algorithme, soit vérifiable sur tout couple de types en temps polynomial.

[2] On se limite également ici au simple cadre des grammaires combinatoires générales algébriques et lexicalisées.

nettement plus aboutis que les nôtres, puisque dans les deux cas un lexique complet et utilisable a été obtenu. Mais nos travaux s'inscrivent dans une perspective très différente des leurs : nous poursuivons l'objectif de respecter les contraintes strictes du modèle de GOLD, qui constituent une garantie de précision très importante. À l'inverse, dans les travaux d'acquisition cités plus haut, diverses techniques propres aux données et/ou à la langue étudiée sont utilisées pour parvenir au résultat. De plus, nos travaux se distinguent également des leurs par l'absence totales d'informations structurées sur les phrases.

Remarque : Ce chapitre reprend nos travaux précédemment publiés dans les articles [77] et [75] (ainsi que dans [76], dans une moindre mesure).

7.2 Apprentissage partiel de grammaires lexicalisées rigides

L'application des algorithmes d'apprentissage de grammaires combinatoires générales rigides se heurte donc au problème classique du rapport entre quantité d'information en entrée et efficacité de l'algorithme : soit l'algorithme est polynomial mais nécessite des $\mathcal{R}$-structures trop complexes, soit l'algorithme n'utilise que des phrases brutes mais est alors exponentiel, donc tout aussi peu utilisable dans le cadre de réelles applications. Précisons que l'on s'intéresse ici uniquement au problème de l'apprentissage sans $\mathcal{R}$-structures, en restant dans le cas des grammaires rigides. Ce second problème sera discuté dans la partie 7.2.4.1.

7.2.1 Méthode

Nous proposons ici un compromis dans lequel les informations que constituent les $\mathcal{R}$-structures (ou FA-structures dans le cas spécifique des grammaires AB) sont remplacées par celles fournies par une *grammaire initiale*, de manière à maintenir un niveau acceptable d'efficacité. Pour des grammaires totalement lexicalisées, telles les grammaires catégorielles, l'hypothèse qu'une partie de la grammaire à apprendre soit connue au départ est réaliste : en effet, cette partie est simplement constituée d'un ensemble de mots auxquels sont associés un ensemble de types.

Afin de montrer combien l'analyse syntaxique et l'apprentissage sont liés dans le cas des grammaires catégorielles, nous proposons ci-dessous (figure 7.1) un petit programme Prolog qui est à la fois un analyseur syntaxique de grammaires AB, un algorithme d'apprentissage de grammaires rigides et un algorithme d'apprentissage partiel. Le point commun entre l'analyse et l'acquisition est la recherche de la structure : dans le premier cas celle-ci garantit l'appartenance de la phrase au langage en utilisant les catégories, et dans le second elle induit les catégories des mots inconnus. Cet algorithme naïf, qui repose entièrement sur le moteur Prolog pour la recherche d'une dérivation et/ou le calcul des types inconnus du lexique, illustre aussi l'importance de l'unification dans le processus de dérivation des grammaires catégorielles.

Bien sûr ce type de programme est extrêmement peu efficace, qu'il soit utilisé comme analyseur ou comme algorithme d'apprentissage : dans les deux cas il effectue d'abord une recherche systématique du parenthésage en constituants de la phrase fournie, avant de vérifier si cette structure peut correspondre

```
:-op(400,xfx,/).
:-op(400,xfx,\).

regle(A/B,B,A).   % Forward Application (FA)
regle(B,B\A,A).   % Backward Application (BA)

couper_liste(Part1, Part2, Liste) :-
        append(Part1, Part2, Liste),
        Part1 \= [],
        Part2 \= [].

deriv(Lexique, [Mot], T) :-
        member(def(Mot,T), Lexique).
deriv(Lexique, Constituant, T) :-
        couper_liste(C1, C2, Constituant),
        deriv(Lexique, C1, T1),
        deriv(Lexique, C2, T2),
        regle(T1,T2,T).

exemple(X) :-    % 'Marie' de type inconnu :
  Lex = [        %  renvoie X = sn
        def('Pierre', sn),
        def('aime', (sn\s)/sn),
        def('Marie', X)
        ],
  deriv(Lex, [ 'Pierre', 'aime', 'Marie' ], s),
  deriv(Lex, [ 'Marie', 'aime', 'Pierre' ], s).
```

Figure 7.1 – Analyse syntaxique et/ou apprentissage naïf en Prolog

(par unification) aux données dont il dispose. On voit cependant qu'il peut aussi bien être utilisé

- comme analyseur : le lexique est entièrement défini, et le prédicat `deriv` termine avec succès si et seulement si la phrase appartient au langage ;
- comme algorithme d'apprentissage traditionnel à partir de simples phrases : tous les types du lexique sont des variables. Le prédicat `deriv` fournira autant de solutions qu'il y a de grammaires compatibles avec les phrases données.
- comme algorithme d'apprentissage partiel (cf. exemple) : une partie des types du lexique sont définis tandis que les autres sont des variables. Le prédicat `deriv` fournira autant de solutions qu'il y a de grammaires compatibles avec les phrases données et le lexique.

On peut noter que cet algorithme termine toujours, car le nombre de parenthésages d'une phrase en constituants (non vides) est borné.

Afin d'optimiser l'utilisation des informations fournie dans la grammaire initiale, on peut procéder de

la même manière que pour l'analyse syntaxique (parsing) : chercher tous les types possibles pour tous les constituants complets (c'est-à-dire ne contenant aucun mot dont le type est inconnu). Selon la proportion de mots inconnus dans la phrase et leur répartition, cela permet de limiter l'explosion combinatoire dûe aux différentes structures possibles. L'algorithme RGPL proposé ci-dessous utilise une méthode de parsing incrémental de type CYK (voir [31],[62],[110]), de manière à guider la construction de l'arbre de dérivation par les types fournis dans la grammaire initiale.

7.2.2 Algorithme

L'algorithme **RGPL** (*Rigid Grammars Partial Learning*) prend en entrée une phrase $w_1, .., w_n$ et une grammaire initiale G_0, composée de règles associant un type à un mot, de la forme $w \triangleright t$. La version présentée ci-dessous (algorithme 7.1) généralise celle présentée dans [75] pour les grammaires AB aux grammaires combinatoires générales : ceci est simplement réalisé à l'aide de la sous-procédure **apply_rules**, dont le rôle est de tenter d'appliquer les règles universelles $\mathcal{R}$ propres à la famille de grammaires considérée, et de renvoyer l'ensemble des possibilités d'applications (couples type et substitutions) le cas échéant. La forme des règles universelles est cependant limitée ici aux règles binaires ($A_1\ A_2\ \rightarrow\ A_0$), puisque nous utilisons un algorithme de type CYK (cette simplification n'est pas indispensable : il suffirait d'utiliser un algorithme basé sur celui d'EARLEY [45] pour supprimer cette limitation).

L'algorithme 7.1 renvoie l'ensemble des grammaires rigides solutions, c'est-à-dire celles contenant les règles de la grammaire initiale et acceptant la phrase $w_1, \ldots, w_n$. L'algorithme 7.2 est un exemple de procédure $apply\_rules$ pour les grammaires AB. Les règles universelles appliquées[3] sont donc les règles FA et BA classiques.

Remarque : dans l'algorithme 7.1, Id désigne la substitution identité. Il est également utile de noter que $(\sigma_u \circ \sigma_l) = (\sigma_u \circ \sigma_r)$ car σ_u est défini comme le MGU de σ_l et σ_r.

Précisons que l'algorithme 7.1 présenté ici ne décrit le fonctionnement que pour une phrase, car il s'agit de la seule partie significative de l'apprentissage partiel. L'algorithme d'apprentissage complet doit simplement utiliser $RGPL$ pour chaque phrase donnée en entrée, puis calculer l'unifieur le plus général de la famille d'ensembles $\{A_1, A_2, \ldots, A_{|\Sigma|}\}$, avec A_i l'ensemble des types associés au mot w_i par l'une des grammaires renvoyées par RGPL. Cette seconde phase est identique à la phase d'unification de l'algorithme RG d'origine (partie 3.2, aussi décrite sous la forme généralisée dans la partie 4.4.2.2).

De même que dans l'algorithme d'apprentissage à partir de $\mathcal{R}$-structures, cet algorithme utilise des termes qui peuvent contenir des variables. Ces variables sont progressivement instanciées selon les contraintes imposées par les règles universelles, en particulier dans le cas où le type est combiné avec un type sans variable. Dans ce but, à chaque étape on vérifie pour chacune des règles s'il existe une substitution permettant d'unifier les types en entrée avec les types de la règle, et le cas échéant on ren-

[3] Il n'est pas difficile de trouver un algorithme générique effectuant cette procédure pour tout ensemble de règles $\mathcal{R}$ (sur le modèle de l'algorithme 4.1, p. 116). Nous omettons cette version plus générale car elle serait évidemment plus complexe, or la généricité n'est pas l'aspect le plus important de la partie présente.

RGPL(G_0**,** $[w_1, w_2, .., w_n]$**)**
[Initialisation]
$Lex \leftarrow \{(W, T) \mid (W \triangleright T) \in G_0\}$
Créer une matrice vide $M[1..n, 1..n]$
Pour i **de** 1 **à** n **faire**
Si ($\exists\, T$ tel que $(w_i, T) \in Lex$) **Alors**
$M[i, i] \leftarrow \{(T, Id)\}$
Sinon
créer une nouvelle variable V
$Lex \leftarrow Lex \cup \{(w_i, V)\}$
$M[i, i] \leftarrow \{(V, Id)\}$
Fin Si
Fin Pour
[Processus de dérivation/apprentissage partiel]
Pour i **de** 2 **à** n **faire**
Pour j **de** $i - 1$ **à** 1 **faire**
Pour k **de** j **à** $i - 1$ **faire**
Pour chaque $(T_l, \sigma_l) \in M[j, k]$ **faire**
Pour chaque $(T_r, \sigma_r) \in M[k + 1, i]$ **faire**
Si ($\exists\, \sigma_u = mgu(\sigma_l, \sigma_r)$) **Alors**
$M[j, i] \leftarrow M[j, i] \cup apply\_rules(T_l, \sigma_l, T_r, \sigma_r, \sigma_u)$
Fin Si
Fin Pour
Fin Pour
Fin Pour
Fin Pour
Fin Pour
[Application des substitutions compatibles]
$Res \leftarrow \emptyset$
Pour chaque $(T, \sigma) \in M[1, n]$ **faire**
Si ($\exists\, \tau$ tel que $\tau(T) = S$) **Alors**
$Res \leftarrow Res \cup \{(\tau \circ \sigma)(Lex)\}$
Fin Si
Fin Pour
Renvoyer Res

ALG. 7.1 – Apprentissage partiel

apply_rules_AB(T_l**,**σ_l**,**T_r**,**σ_r**,**σ_u**)**
$Res \leftarrow \emptyset$
Créer deux nouvelles variables A, B
Si ($\exists\, \sigma_{FA} = mgu(\{\{\sigma_u(T_l), A/B\},\ \{\sigma_u(T_r), B\}\})$) **Alors**
  $Res \leftarrow Res \cup \{(\sigma_{FA}(A), \sigma_{FA} \circ \sigma_u \circ \sigma_l\}$
Fin Si
Créer deux nouvelles variables A', B'
Si ($\exists\, \sigma_{BA} = mgu(\{\{\sigma_u(T_l), B'\},\ \{\sigma_u(T_r), B'\backslash A'\}\})$) **Alors**
  $Res \leftarrow Res \cup \{(\sigma_{BA}(A'), \sigma_{BA} \circ \sigma_u \circ \sigma_l\}$
Fin Si
Renvoyer Res

ALG. 7.2 – Apprentissage partiel, sous-procédure $apply\_rules\_ab$

voie le type résultant de cette substitution et la composition de cette substitution avec les précédentes. Chaque type "réalisable" par un constituant est accompagné de la substitution qui permet de l'obtenir. Cette substitution porte sur les variables associées aux mots inconnus de ce constituant, afin de vérifier lors de chaque combinaison de types que leurs substitutions sont compatibles (par calcul du MGU). Bien sûr, à la différence du cas de l'apprentissage avec structures, il est possible (et fréquent) qu'une variable ne soit pas totalement instanciée, ou que plusieurs instanciations soient possibles.

Il est important de noter que malgré son fonctionnement "de type CYK" la complexité de cet algorithme n'est évidemment pas polynomiale en général. En effet, le nombre de couples (T, σ) dans chaque case de la matrice M peut croître de manière exponentielle. C'est notamment le cas lorsque tous les mots sont inconnus (la grammaire initiale est vide) : on se trouve alors dans le cas précédent d'apprentissage à partir de simples phrases, et l'algorithme doit calculer l'ensemble des structures possibles.

7.2.3 Exemple

Soit G_0 la grammaire initiale définie par le lexique suivant[4]

$$\{\ un : SN/N,\ homme : N,\ poisson : N,\ nage : SN\backslash S,\ vite : (SN\backslash S)\backslash(SN\backslash S)\ \}.$$

Soit "$un\ homme\ court$" la phrase donnée comme entrée à l'algorithme, où "$court$" est un mot inconnu.

1. *Initialisation :*
   $M[1,1] \leftarrow (SN/N, \emptyset)$
   $M[2,2] \leftarrow (N, \emptyset)$
   $M[3,3] \leftarrow (x_1, \emptyset)$ et $Lex \leftarrow Lex \cup \{(court, x_1)\}$

[4]On peut noter dans cette grammaire que le type des noms N est un argument du type du déterminant SN/N, et non l'inverse : il s'agit de la notation classique dans les grammaires catégorielles.

2. *Dérivation :*

$i = 2, j = 1, k = 1 :$ $M[1,2] \leftarrow (SN, \emptyset)$ (FA)
$i = 3, j = 2, k = 2 :$ $M[2,3] \leftarrow (x_2, \{x_1 \mapsto N \backslash x_2\})$ (BA)
$i = 3, j = 1, k = 1 :$ $M[1,3] \leftarrow (SN, \{x_1 \mapsto N \backslash N\})$ (FA)
$i = 3, j = 1, k = 1 :$ $M[1,3] \leftarrow (x_3, \{x_1 \mapsto N \backslash ((SN/N) \backslash x_3)\})$ (BA)
$i = 3, j = 1, k = 2 :$ $M[1,3] \leftarrow (x_4, \{x_1 \mapsto SN \backslash x_4)\})$ (BA)

3. *Substitutions compatibles avec S :*

$\tau(SN) = S$? impossible
$\tau(x_3) = S$? $\{x_1 \mapsto N \backslash ((SN/N) \backslash S)\}$
$\tau(x_4) = S$? $\{x_1 \mapsto SN \backslash S\}$

Après cet exemple, il y a deux grammaires dans l'ensemble des solutions : l'une définit "$court$" par le type $N \backslash ((SN/N) \backslash S)$, l'autre par le type $SN \backslash S$. Si l'exemple "$un\ homme\ court\ vite$" apparaît plus tard dans la séquence d'exemples, la première solution sera éliminée, car il n'existe pas de substitution sur les règles universelles permettant de combiner $N \backslash ((SN/N) \backslash S)$ avec le type de l'adverbe $vite$, $(SN \backslash S) \backslash (SN \backslash S)$.

7.2.4 Discussion

L'algorithme RGPL, malgré son apparence très différente, est très proche dans le principe de l'algorithme RG de BUSZKOWSKI (ou de l'algorithme 4.4 dans la version généralisée). Dans les deux cas, tout repose sur le fait que les règles universelles s'appliquent uniquement si une substitution existe. Ceci permet de rechercher les solutions en calculant à chaque pas un unifieur le plus général, ce qui garantit à la fois l'adéquation des solutions trouvées aux informations disponibles (la solution est la plus précise possible par rapport aux données), et l'absence de surgénéralisation (la solution ne sera jamais en contradiction avec une donnée pas encore disponible). C'est aussi en grande partie ce fonctionnement basé sur l'unification qui rend possible l'apprentissage partiel, puisqu'il autorise la présence simultanée de variables instanciées et non instanciées. Bien entendu, le principal problème demeure l'absence de garantie d'efficacité de ce type d'algorithme, puisque celle-ci dépend totalement de la proportion et de la répartition des mots connus et inconnus.

7.2.4.1 Contrainte de rigidité et grammaire minimale

L'algorithme RGPL présenté ci-dessus apprend uniquement des grammaires rigides. Plus exactement, les mots définis dans la grammaire initiale pourraient avoir plusieurs catégories, mais l'algorithme unifie tous les types d'un même mot inconnu. Dans le cas des grammaires AB, il serait possible de gérer des grammaires k-valuées, mais cela serait contraire à l'objectif puisque l'algorithme deviendrait très rapidement inexploitable : il serait en effet nécessaire de recourir à la méthode proposée par KANAZAWA, qui consiste à calculer toutes les possibilités d'unifier ou non chaque couple de types d'un même mot (voir partie 3.3.2). Dans le même ordre d'idée, nous n'apportons pas non plus ici de solution au problème du calcul (efficace) d'une grammaire minimale parmi l'ensemble des solutions.

On peut par contre envisager pour pallier ces problèmes que des classes (syntaxiques) de mots soit définies dans la grammaire initiale : chaque mot inconnu devrait alors appartenir à l'une des classes prédéfinies, ce qui limite fortement les combinaisons de types. Le regroupemement par classes est fréquemment utilisé (par exemple dans l'étiqueteur de BRILL [25]), mais suppose que les classes prédéfinies couvrent l'ensemble des combinaisons syntaxiques suceptibles d'apparaître dans les exemples, et ne causent pas de surgénération. Même si elle présente un indéniable intérêt pratique, cette solution est contestable sur le plan théorique puisqu'alors l'algorithme ne fait plus véritablement d'inférence grammaticale au sens strict du modèle de GOLD.

7.2.4.2 La grammaire initiale

L'intérêt de la méthode d'apprentissage que nous proposons repose entièrement sur l'existence d'une grammaire initiale. Celle-ci doit être suffisamment complète pour qu'un nombre significatif de mots soient connus dans les phrases de la séquence à apprendre. Dans ce cadre, on peut tirer profit de la "loi de ZIPF", utilisée notamment par l'étiqueteur de BRILL [25]. Celle-ci garantit que, sur l'ensemble des mots d'un texte, une faible proportion des mots suffit à représenter une grande partie du texte (en nombre d'occurences). Or précisément ce sont les mots les plus fréquents d'un langage qui sont le plus facile à répertorier et définir (mots grammaticaux tels que déterminants, pronoms, prépositions, conjonctions, etc.), soit manuellement soit par conversion de dictionnaires existants dans d'autres formalismes. Nous proposons dans la partie 7.3 quelques tests qui mettent en évidence l'intérêt de la loi de ZIPF pour l'apprentissage partiel.

7.3 Expérimentation avec les grammaires de liens

Les premiers tests de l'apprentissage partiel présentés dans cette partie utilisent le lexique des grammaires de liens pour l'anglais. Cette grammaire a été construite par Sleator et Temperley [106] et offre une description assez précise et complète (au niveau grammatical) de l'anglais. L'apprenabilité des grammaires de liens k-valuées a été démontrée par Béchet dans [14], et nous en avons donné une nouvelle démonstration dans la partie 5.3.3.3. Dans notre cas, les grammaires de liens ont surtout l'avantage d'être très proches des grammaires catégorielles, ce qui rend l'adaptation de l'algorithme d'apprentissage partiel relativement aisée. Nous avons en effet proposé dans la partie 4.3.4 une formalisation des grammaires de liens sous forme de grammaires combinatoires générales, nommée grammaires catégorielles de liens. Malgré les limitations qui en découlent (cycles non représentables, voir partie 4.3.4), ce formalisme intermédiaire reste relativement fidèle à celui d'origine.

7.3.1 Analyse syntaxique simple pour les grammaires catégorielles de liens

Le fait que les auteurs Sleator et Temperley aient construit un lexique de la langue anglaise dans le formalisme des grammaires de liens constitue un avantage pratique non négligeable. Ce lexique contient approximativement 59000 mots, répartis dans 1350 classes syntaxiques, chaque classe correspondant à un ensemble de types. La grammaire gère une grande partie des phénomènes linguistiques de l'anglais,

ce qui selon les auteurs “indique que cette approche peut avoir des utilisations pratiques et être pertinente au niveau linguistique” [98]. De plus l’analyseur syntaxique fourni par les auteurs inclut un fichier d’exemples de 928 phrases, étiquetées comme correctes ou incorrectes.

Nous décrivons ci-dessous comment le lexique d’origine a été converti dans le formalisme des grammaires catégorielles de liens. Cette conversion prend en compte certaines des “caractéristiques avancées” qui ont été ajoutées par les auteurs au système de base des grammaires de liens. En effet, l’application réelle des grammaires de liens à l’analyse de la langue anglaise a requis plusieurs adaptations techniques par rapport à la stricte définition formelle de celles-ci ([98], [52]). C’est pourquoi nous avons développé dans un premier temps un analyseur syntaxique de grammaires catégorielles de liens (GCL), dans le but de vérifier la fiabilité de la conversion que nous proposons du lexique d’origine en GCL. Dans la mesure où l’objectif final n’est pas la réalisation de cet analyseur, plusieurs simplifications ont été effectuées, ce qui aboutit à quelques limitations pratiques détaillées ci-dessous. Ces choix sont principalement guidés par la nécessité de préserver la simplicité du système, de façon à ce qu’il reste le plus proche possible du système de base décrit dans la partie 4.3.4. Dans le cadre de l’apprentissage partiel, il faut en effet que certaines propriétés soient maintenues, telles que la possibilité de retrouver une partie de dérivation uniquement par le mécanisme de l’unification. De plus, le fait de préserver un niveau de généralisation assez élevé rend le système plus souple pour d’éventuelles futures améliorations ou nouvelles utilisations.

7.3.1.1 Conversion des catégories au format du dictionnaire

Dans le fichier dictionnaire fourni par les auteurs, les entrées ont le format (simplifié) suivant :

$< liste\_mots >::=< Categ >$,

avec $< liste\_mots >$ une liste d’entrées lexicales séparés par des espaces, et $< Categ >$ un terme dont les atomes sont les connecteurs (éventuellement précédé de l’opérateur `@`, cf. ci-dessous), et les opérateurs sont les opérateurs logiques `and` et `or`. Les connecteurs à droite (resp. à gauche) sont notés `C+` (resp. `C-`), et l’ordre n’est important que pour les connecteurs d’un même côté (i.e. on peut avoir aussi bien “`A- and B- and C+`” que “`A- and C+ and B-`”). Un opérateur d’optionnalité, noté “`?`” est aussi utilisé : il sert à indiquer que le sous-terme de la catégorie qu’il inclut peut être utilisé, mais n’est pas obligatoire[5]. Enfin les multi-connecteurs, notés par l’opérateur “`@`”, permettent de brancher un nombre indéfini de liens sur un même connecteur : un mot défini avec la catégorie `@A+` peut être connecté à un ou plusieurs mots à droites par leur connecteur `A-`. Par exemple, les noms peuvent avoir le type `d([@A,D],[S])`, de manière à autoriser qu’un nombre quelconque d’adjectifs précède le nom.

Le convertisseur implémenté met simplement les catégories sous forme normale disjonctive :

[5] Remarque : cet opérateur n’est bien-sûr jamais utilisé à la racine de la catégorie, ce qui autoriserait le mot à apparaître à n’importe quelle position d’une phrase.

$$
\begin{array}{ll}
conv(\texttt{A-}) & = \{\, \texttt{d([A] , [ ])} \,\} \\
conv(\texttt{A+}) & = \{\, \texttt{d([ ] , [A])} \,\} \\
conv(\texttt{@A-}) & = \{\, \texttt{d([@A] , [ ])} \,\} \\
conv(\texttt{@A+}) & = \{\, \texttt{d([ ] , [@A])} \,\} \\
conv(\texttt{A or B}) & = conv(\texttt{A}) \cup conv(\texttt{B}) \\
conv(\texttt{A and B}) & = \{\, \texttt{d(L•L' , R•R')} \mid \texttt{d(L,R)} \in conv(\texttt{A}) \text{ et } \texttt{d(L',R')} \in conv(\texttt{B}) \,\} \\
conv(\texttt{A?}) & = conv(\texttt{A}) \cup \{\, \texttt{d([ ] , [ ])} \,\}
\end{array}
$$

Remarque : cette mise sous forme normale disjonctive implique nécessairement une explosion combinatoire (produit cartésien dans le cas des conjonctions). Néanmoins cette opération ne porte que sur la grammaire et n'a donc pas d'effet sur la complexité de l'analyse syntaxique en elle-même[6].

La possibilité d'itérations sur les connecteurs, représentée à l'aide des *multi-connecteurs*, est maintenue dans le formalisme des GCL, en remplaçant les règles de dérivation "de base" par les suivantes (dans lesquelles `[@]` signifie que le symbole `@` est optionnel) :

```
    d(L, cons([@]c, R)) , d(cons([@]c, nil), nil) → d(L,R)
    d(nil, cons([@]c, nil)) , d(cons([@]c, L), R) → d(L,R)
d(L, cons(@c, R)) , d(cons([@]c, nil), nil) → d(L,cons(@c, R))
d(nil, cons([@]c, nil)) , d(cons(@c, L), R) → d(cons(@c, L),R)
```

7.3.1.2 Subscripts

Les connecteurs utilisés dans la grammaire originale peuvent être dotés de *subscripts*. Les *subscripts* sont utilisés pour spécialiser les connecteurs, d'une manière similaire aux structures de traits dans les grammaires d'unification. Ils sont utilisés pour rendre la grammaire plus facile à comprendre et éventuellement à modifier, et cela diminue le nombre de connecteurs nécessaires à la description du langage. Par exemple, les subscripts `s` et `p` sont utilisés avec les noms (ainsi que les pronoms, déterminants, etc.) pour distinguer ceux qui sont au singulier de ceux qui sont au pluriel[7].

Pour maintenir la simplicité des règles de réduction, les types d'origine utilisant ce mécanisme sont convertis en types sans subscripts. Ceci est réalisé à l'aide d'un programme qui classifie tous les connecteurs existants (avec leurs subscripts) de telle sorte que tous les connecteurs d'une même classe ne puisse être reliés qu'avec le même ensemble de connecteurs. Afin de minimiser le nombre de classes, deux classes sont fusionnées chaque fois que cela est possible. A la fin du processus un nouveau type est créé pour chaque classe : le lexique ainsi converti est équivalent au lexique d'origine. Soulignons qu'une telle conversion est purement formelle : par exemple si plusieurs subscripts `+` sont compatibles avec exactement le même ensemble de subscripts `-`, ceux-ci sont unifiés et considérés comme un même subscript. L'intérêt d'un tel fonctionnement est d'éviter de produire un trop grand nombre de connecteurs, ce qui

[6] L'étape de conversion étant bien entendu distincte de celle de l'analyse utilisant le lexique converti.

[7] En pratique, le subscript d'un connecteur est la partie (éventuellement vide) écrite en minuscule, comme *s* dans *Ds+*. Les connecteurs peuvent avoir plusieurs caractères de subscript, comme *Spa+*. Un *Spa+* peut se brancher avec un *S-*, un *Sp-*, ou un *Spa-*, mais pas avec un *Ss-*, ni un *Ssa-* ou un *Spb-*.

amplifierait encore l'explosion combinatoire dans la conversion des catégories. Cependant cela a pour inconvénient la perte du sens associé à chaque subscript dans le dictionnaire d'origine.

7.3.1.3 Analyse lexicale (segmentation en mots)

Pour la segmentation en mots (*tokenization*), les critères utilisés par SLEATOR et TEMPERLEY sont suivis, notamment en ce qui concerne la ponctuation, les nombres, les mots avec trait d'union (*hyphenated words*) ou majuscule (*Capitalized words*), et les expressions figées (que les auteurs nomment *idioms*). Cette étape est réalisée avant l'analyse syntaxique proprement dite. Toutefois certains aspects liés à la segmentation en mots sont gérés par l'analyseur : affectation des "catégories spéciales" pour les mots avec majuscule, traits d'union, nombres. La gestion des "*walls*" (optionnelle) est maintenue : les "*walls*" sont deux éléments lexicaux spéciaux placés avant et après la phrase, qui doivent satisfaire certaines connexions. A noter que cette caractéristique, héritée du système original, est très utile pour construire un arbre de dérivation, puisqu'elle permet d'en retrouver la racine.

En revanche, la gestion des mots inconnus est abandonnée à ce niveau, puisque l'objectif à terme est précisément de traiter de mots inconnus dans un cadre d'apprentissage.

7.3.1.4 Fonctionnalités non gérées et implications sur l'échantillon d'exemples

Selon les auteurs, *"Les constructions utilisant des coordinations ne s'intègrent pas naturellement dans le cadre des grammaires de liens.*" [106]. Ainsi, l'analyseur original dispose d'un système spécifique pour gérer les conjonctions. Le prototype CLG implémenté ne permet pas d'en tenir compte actuellement, c'est pourquoi les phrases contenant des conjonctions son supprimées du fichier d'exemples. Ceci concerne une cinquantaine de phrases destinées spécifiquement à tester cet aspect. Une douzaine d'autres phrases contenant des conjonctions sont séparées en plusieurs phrases sans conjonctions. Par exemple, *"The former astronaut was alone and afraid"* devient *"The former astronaut was afraid"* d'une part et *"The former astronaut was alone"* d'autre part. Sur les 928 phrases de l'échantillon d'origine fourni avec l'analyseur, 892 sont conservées après ce traitement.

L'analyseur d'origine effectue un traitement simplement nommé "*post-processing*" par les auteurs, réalisé après l'analyse de la phrase proprement dite (comme son nom l'indique). Ce traitement vérifie que la structure de liens trouvée satisfait certaines règles de formation, basée sur le type de connecteurs utilisés. Il semble qu'un grand nombre de phrases éliminées à cette étape (donc considérées incorrectes) le soient pour des raisons plus sémantiques que syntaxiques. Par exemple, la phrase "*The question is who we should invite*" est correcte, tandis que "**The party is who we should invite*" ne l'est pas d'après les règles de post-processing. Comme ce système de post-processing est difficilement convertible au cadre formel des CLG d'une part, et que d'autre part les aspects sémantiques sont secondaires dans l'optique de l'apprentissage de règles syntaxiques, cet aspect est abandonné dans notre analyseur. En conséquence, les 116 phrases incorrectes nécessitant cette étape sont retirées du jeu d'essai (il en reste 776).

Les auteurs ont également mis en place un système qui affecte un coût à chaque lien, afin à la fois d'accélérer la recherche d'une structure de liens et d'éliminer certaines constructions syntaxiques

dont la probabilité est faible. Cette heuristique ne peut être exportée dans le cadre de l'apprentissage symbolique, c'est pourquoi nous n'en avons pas tenu compte dans l'implémentation de notre analyseur. Sur l'ensemble de l'échantillon d'exemples d'origine, seulement 13 phrases incorrectes nécessitent que le système de coût soit utilisé (sur 928, soit 1,4%). Parmi les 776 phrases du jeu d'essai obtenu après les étapes précédentes il n'y en a que 5. Il reste donc 771 phrases dans l'échantillon.

Le parser original dispose d'un traitement pour les mots ne figurant pas dans le dictionnaire. Une seule phrase dans le fichier d'exemples est concernée (utilisation du mot "*subjetc*" au lieu de "*subject*"), celle-ci est conservée mais étiquetée incorrecte.

7.3.1.5 Le problème des cycles

On a vu que le système de base que nous proposons était équivalent au système de base des grammaires de liens, mais uniquement dans le cas où la structure de liens ne contient pas de cycles (voir partie 4.3.4). Il s'agit là de la principale différence d'approche entre le système de SLEATOR et TEMPERLEY et le nôtre.

Cette restriction est dûe à la contrainte posée sur les règles des CLG : celles-ci doivent être exprimées sous forme de règles de réécritures entre deux séquences de termes, afin d'être utilisables dans un cadre d'apprentissage formel. Sur le plan pratique, les cas d'utilisation des cycles dans les grammaires de liens sont heureusement assez précisément définis. En effet plusieurs connecteurs doivent obligatoirement être utilisés en produisant un cycle (pour éviter certaines erreurs syntaxiques, cf. [106]). Ces connecteurs sont identifiés, ce qui nous a permis de les éliminer du lexique. Cette opération est possible puisque, comme ces connecteurs n'apparaissent que dans des cycles, il y a nécessairement un ou plusieurs autres connecteurs dans le même type.

Cependant les connecteurs supprimés provoqueront des erreurs dans le cas de phrases incorrectes : en effet l'absence des connecteurs de cycles permet toujours de construire une structure de phrase sans cycle, mais n'empêche plus une telle construction lorsque celle-ci était impossible avec la gestion des cycles. Il s'agit d'un inconvénient important pour lequel aucune solution pratique n'est encore réalisée[8]. Le fichier d'exemples constitué comme décrit ci-dessus contient 771 phrases, dont 550 correctes et 221 incorrectes. L'absence de gestion des cycles dans l'analyseur de CLG provoque 38 erreurs (4,9%). Ces erreurs ne concernent que des phrases incorrectes : cela semble montrer que les connecteurs "classiques" définis dans ce lexique de l'anglais ne doivent jamais produire de cycles.

7.3.1.6 L'analyseur syntaxique de GCL

La correction de la conversion a été testée en comparant les résultats de l'analyse syntaxique des phrases du fichier d'exemples avec ceux obtenus par l'analyseur fourni par les auteurs. L'analyseur de GCL est un analyseur standard de type CYK, programmé en SWI Prolog (et non optimisé), qui applique

[8] Afin de mesurer l'importance des cycles dans le système de liens original, le parser CLG dispose d'une option leur gestion. Si celle-ci est activée, des règles de dérivation spéciales sont utilisées de manière à simuler le même type de fonctionnement que l'analyseur d'origine. Mais ces règles ne sont pas exploitables dans le cadre de l'apprentissage partiel.

simplement les règles de réduction binaires. Conformément aux explications données plus haut, cet analyseur ne gère pas certaines des fonctionnalités avancées de l'analyseur d'origine, ce qui implique la suppression de l'échantillon des phrases nécessitant ces fonctionnalités pour être traitées correctement. On obtient donc un échantillon de 771 phrases parmi lesquelles 221 sont incorrectes. Sur cet ensemble de phrases, l'analyseur de GCL obtient exactement les mêmes résultats[9] que l'analyseur d'origine.

Exemple 7.1. Voici quelques phrases contenues dans le fichier d'exemples, celles qui sont incorrectes étant précédées d'une astérisque :

What did John say he thought you should do
**What did John say did he think you should do*
To pretend that our program is usable in its current form would be silly
That our program will be immediately accepted is hardly likely
**Is that our program will be accepted likely*
The man there was an attempt to kill died

7.3.2 Expérimentations et résultats

L'objectif des expérimentations présentées ci-dessous est de tester la faisabilité en pratique de l'apprentissage partiel avec les données issues du lexique anglais des grammaires de liens. Le prototype utilisé est codé en SWI Prolog. Il est important de noter que la seule partie du problème observée dans ces tests est la possibilité de réaliser l'étape de construction des grammaires générales par apprentissage partiel en un temps raisonnable : aucune phase d'unification ni de calcul de la grammaire minimale n'est réalisée (voir discussion dans la partie 7.2.4.1).

7.3.2.1 Le prototype d'apprentissage partiel des GCL

Le prototype d'apprentissage partiel est basé sur l'analyseur syntaxique de grammaires catégorielles de liens présenté en 7.3.1, et utilise l'algorithme 7.1. Le langage Prolog est de toute évidence bien adapté pour manipuler des termes contenant des variables au cours du processus : en pratique les substitutions ne sont pas stockées dans la structure de données comme indiqué dans l'algorithme, mais simplement obtenues à l'aide du mécanisme d'unification du langage Prolog. L'analyse demeure néanmoins déterministe : les types sont copiés chaque fois que c'est nécessaire de manière à maintenir des listes de types dans la matrice, et ainsi éviter le *backtracking* sur la matrice.

En dehors de quelques optimisations non essentielles, le programme a aussi une caractéristique importante qui permet d'éviter une part conséquente de l'explosion combinatoire. Il s'agit de la factorisation des types "structurellement équivalents", sachant que deux types t et t' sont *structurellement équivalents* s'il existe deux substitutions σ_1 et σ_2 telles que $\sigma_1(t) = t'$ et $\sigma_2(t') = t$ (en d'autres termes les deux types sont identiques modulo un renommage de variables). Dans la mesure où il est fréquent que deux types structurellement équivalents appartiennent à la même case de la matrice, cette optimisation diminue de manière significative le temps d'exécution et l'espace mémoire utilisé. Cependant ce mécanisme

[9] Exception faite de la durée des calculs, beaucoup plus longue pour notre analyseur.

nécessite que la composition des types ne soit plus calculée à chaque étape au cours de l'analyse, sinon il faudrait redévelopper tous les types dès qu'on passe au niveau suivant. C'est pourquoi seule la substitution utilisée pour transformer un type à un niveau est conservée. Le programme doit donc réaliser une seconde phase qui calcule les compositions globales des substitutions, après la phase d'analyse incrémentale.

Les tests sont faits à partir du fichier d'exemples fourni avec l'analyseur syntaxique des grammaires de liens [106]. Seule une partie des phrases correctes de cet ensemble d'exemples a été conservée. La segmentation en mots des phrases est réalisée (et vérifiée manuellement) avant le processus d'apprentissage partiel. Les tests consistent à supprimer une partie des mots du lexique, de manière à ce que ces mots soient considérés comme inconnus.

7.3.2.2 Résultats

L'échantillon utilisé contient 537 phrases[10] et 4765 mots (taille de l'échantillon), dont 1064 mots différents (taille du lexique). Les mots les plus fréquents sont *the* (270), *I* (155), *is* (122), *to* (121). Dans ces tests nous comparons les temps de calcul de l'apprentissage partiel pour différents taux de mots inconnus (dans le lexique et dans l'échantillon).

Dans les résultats ci-dessous l'abbréviation "MI" désigne les mots inconnus. La première colonne (lexique) indique le nombre de mots inconnus parmi l'ensemble des mots, tandis que la seconde colonne (échantillon) indique le nombre d'occurences de ces mots inconnus dans les phrases. Dans ce premier test les mots supprimés du lexique sont choisis aléatoirement. Les résultats sont présentés dans le tableau 7.2. On peut voir que le taux de mots inconnus dans l'échantillon est proche de celui dans le lexique.

MI (lexique)	MI (échantillon)	Temps total	MI par phrase (min ; moy ; max)	temps par phrase (min ; moy ; max) (sec)
0 (0%)	0 (0%)	27 min	0 ; 0 ; 0	0,1 ; 3,1 ; 14,7
106 (10%)	407 (8,5%)	75 min	0 ; 0,7 ; 4	0,1 ; 8,3 ; 333
211 (20%)	870 (18,2%)	3,5 h	0 ; 1,5 ; 6	0,1 ; 24 ; 1985
264 (25%)	1106 (23,2%)	3,8 h	0 ; 1,8 ; 7	0,1 ; 24,9 ; 1115
317 (30%)	1481 (31,1%)	Échec (mémoire insuffisante)		

Figure 7.2 – Suppression aléatoire de mots du lexique.

Afin de simuler l'idée qu'il est plus judicieux de construire la grammaire initiale en utilisant un petit ensemble de mots (très) fréquents, seuls les mots les moins fréquents sont supprimés du lexique dans

[10] Bien sûr seules les phrases correctes sont prises en compte pour l'apprentissage. Sur les 550 présentes dans l'échantillon (voir partie 7.3.1), 13 ont dues être supprimées : la plupart l'ont été soit parce qu'elles étaient très longues, soit parce qu'elles contenaient trop souvent une grande quantité de mots inconnus, ce qui provoquait l'échec du processus par saturation de la mémoire. Il s'agit bien sûr d'une limitation sérieuse du principe de l'apprentissage partiel.

ce second test. Grâce à la loi de ZIPF, il est ainsi possible de supprimer une grande partie des mots du lexique sans avoir trop de mots inconnus dans l'échantillon, comme le montre le tableau 7.3.

MI (lexique)	MI (échantillon)	Temps total	MI par phrase (min ; moy ; max)	temps par phrase (min ; moy ; max) (sec)
0 (0%)	0 (0%)	27 min	0 ; 0 ; 0	0,1 ; 3,1 ; 14,7
211 (20%)	211 (4,4%)	45 min	0 ; 0,3 ; 4	0,1 ; 5,0 ; 173
422 (40%)	422 (8,8%)	78 min	0 ; 0,6 ; 5	0,1 ; 8,7 ; 620
634 (60%)	727 (15,2%)	2,6 h	0 ; 1,1 ; 7	0,1 ; 17,3 ; 1844
845 (80%)	1302 (27,3%)	5,1 h	0 ; 2,0 ; 9	0,1 ; 34,1 ; 1893

Figure 7.3 – Suppression du lexique des mots les moins fréquents.

Ces tests montrent que le processus d'apprentissage partiel peut fonctionner en un temps acceptable (et sans surcharger l'espace mémoire) avec un taux de mots inconnus dans l'échantillon allant jusqu'à 25 à 30%. On peut constater qu'il suffit qu'environ 20% des mots du lexique soient définis pour obtenir un tel taux, grâce à la loi de ZIPF.

Bien entendu, les données sur lesquelles sont réalisées ces tests ne sont pas suffisamment représentatives pour garantir que l'algorithme serait en mesure de fonctionner dans toute situation où le taux de mots inconnus serait celui-ci. Mais le lexique anglais des grammaires de liens, qui est une approximation raisonnable d'un langage naturel, n'était pas conçu pour ce type d'application. Par conséquent ces tests semblent montrer que la méthode de l'apprentissage partiel pourrait être une approche réaliste pour appliquer l'apprentissage symbolique aux langues naturelles.

7.4 Conclusion et perspectives

Dans le domaine de l'inférence grammaticale, l'apprenabilité des classes de langages est souvent étudiée indépendamment d'une éventuelle mise en pratique de cet apprentissage. Du strict point de vue de l'apprenabilité dans le modèle de GOLD, aucune distinction n'est nécessaire entre les cas d'apprentissage de grammaires rigides à partir de FA-structures et de grammaires k-valuées à partir de simples phrases, même si le premier est peu utile et le second pratiquement irréalisable.

Dans la perspective d'applications "réelles", nous avons proposé une adaptation des algorithmes d'apprentissage de grammaires catégorielles. L'approche étudiée est moins contraignante sur la forme des entrées de l'algorithme, tout en restant relativement réaliste du point de vue de l'efficacité. Elle est basée sur la lexicalisation totale et l'unification, ce qui en fait une méthode potentiellement utilisable avec de nombreux formalismes. Néanmoins il reste plusieurs questions à résoudre avant de pouvoir appliquer des algorithmes d'apprentissage symbolique de règles syntaxiques au langage naturel : comme on l'a vu, le passage aux grammaires k-valuées, indispensable pour garantir une expressivité suffisante des classes

de langages concernées, reste une source d'inefficacité des algorithmes qui semble rédhibitoire. Il reste également à déterminer comment et selon quels critères constituer la grammaire initiale.

Sur un plan pratique, on peut s'interroger sur ce qu'un algorithme d'apprentissage doit renvoyer comme résultat : dans l'idéal ce devrait être une unique grammaire, mais le calcul de la grammaire minimale parmi un ensemble de grammaires solutions est définitivement trop complexe, quand il n'est pas impossible. Même si l'on tolère que la réponse de l'algorithme soit un ensemble de grammaires, cet ensemble risque souvent d'être beaucoup trop grand pour être calculable. On serait alors tenté d'utiliser des représentations plus flexibles pour les types, comme dans la version originale des grammaires de liens où les types sont des formules contenant des opérateurs logiques (*et*, *ou*, etc.) ce qui permet de factoriser les similarités (donc représenter plusieurs grammaires en une seule). Mais l'apprentissage symbolique a besoin de pouvoir reconnaître lorsque deux types sont identiques, ce qui suppose que leur représentation soit normalisée.

Globalement, il semble donc très difficile de respecter toutes les contraintes imposées par le modèle de GOLD dans le cadre d'un apprentissage automatique réel (sur de grandes quantités de données) appliqué au langage naturel. Cependant il ne faut sans doute pas en déduire trop vite que les obstacles sont insurmontables : malgré le pessimisme initial qui a longtemps pesé sur ce modèle, il s'est avéré en mesure de représenter des distinctions non évidentes entre classes apprenables et non apprenables. Mais même s'il est nécessaire de sortir du cadre strict du modèle de GOLD pour trouver des applications aux algorithmes d'apprentissage qui en découlent, par exemple en utilisant des types fixés dans des classes de mots prédéfinis, l'apprentissage symbolique a principalement un avantage, à savoir la précision. En effet, cette forme d'apprentissage garantit une précision maximale qui peut être un atout important dans certaines tâches, même s'il est probable que l'efficacité en restera le principal défaut.

Conclusion et perspectives

Après avoir été longtemps considéré comme trop restrictif, les travaux d'ANGLUIN et ceux qui les ont suivis ont permis de découvrir en quoi le modèle de GOLD est pertinent. Cependant, les études réalisées dans ce cadre sont essentiellement restées à un niveau d'abstraction élevé, assez éloigné des formalismes grammaticaux potentiellement utiles aux langues naturelles. Le niveau d'abstraction auquel on a étudié le modèle de GOLD est progressivement descendu, notamment à travers les résultats de WRIGHT (l'élasticité finie), de SHINOHARA (la densité finie bornée, l'apprenabilité des grammaires contextuelles k-bornées) et de KANAZAWA (les diverses applications aux grammaires catégorielles).

Comme point de départ de notre étude, nous nous sommes attachés plus particulièrement aux travaux de KANAZAWA [60] sur l'apprentissage des grammaires catégorielles classiques dans le modèle de GOLD. Ceux-ci constituent en effet de premiers résultats très intéressants sur un formalisme grammatical lié aux langues naturelles. Nous avons vu notamment qu'un algorithme d'apprentissage efficace existait pour le cas des langages de structures de grammaires AB rigides, et que les langages de chaînes des grammaires AB k-valuées sont apprenables. Dans la perspective d'applications aux langues naturelles, nous avons constaté que cela présente principalement deux problèmes : d'une part la pauvreté structurelle des grammaires AB sur le plan linguistique nécessite la recherche de formalismes plus adaptés, et d'autre part la mise en œuvre de l'apprentissage sur des données de taille importante est rendue difficile par la complexité algorithmique des méthodes existantes.

Concernant l'extension de l'apprenabilité à des formalismes grammaticaux plus complexes que les grammaires AB, nous avons choisi d'utiliser le cadre des grammaires combinatoires générales, dans une version étendue par rapport à la définition qu'en donne KANAZAWA. Nous avons vu qu'un certain nombre de formalismes, plus ou moins proches des grammaires catégorielles, sont représentables dans ce modèle (de façon relativement fidèle). Simultanément, cela offre un niveau de généralisation suffisamment élevé pour envisager l'application des résultats d'apprenabilité obtenus à d'autres formalismes que ceux étudiés ici. Nous avons démontré aussi que le modèle possède certaines propriétés favorables à l'apprenabilité, notamment dans le cas des langages de structures de grammaires rigides.

La restriction de l'apprenabilité aux classes de langages de grammaires rigides limite fortement l'expressivité de celles-ci. Afin d'assurer aux classes de langages dont nous étudions l'apprenabilité l'expressivité nécessaire aux langues naturelles, il est donc indispensable de considérer le cas des grammaires

k-valuées (ou k-bornées). En nous inspirant des travaux de KANAZAWA, nous avons ainsi recherché des conditions suffisantes à l'élasticité finie des classes de grammaires combinatoires générales. Dans ce but, nous avons proposé une généralisation de la notion de densité finie bornée définie par SHINOHARA [95], ce qui permet notamment de rattacher l'apprenabilité des grammaires AB à ce cadre (et donc de souligner la convergence des méthodes de preuve d'apprenabilité). Les deux critères proposés portent sur les règles universelles définies pour la famille de grammaires, ce qui facilite leur vérification pour un formalisme donné. Le premier de nos résultats démontre que les classes de langages de structures de grammaires (lexicalisées) k-bornées à arguments bornés ont l'élasticité finie. Ce critère a l'avantage d'être assez souple (donc d'inclure une grande variété de formalismes), mais restreint la forme des grammaires (par une borne sur la taille des types arguments). Le second résultat démontre l'élasticité finie des classes de langages de structures de grammaires (algébriques et lexicalisées) k-bornées par consommation stricte d'arguments. Il s'agit à l'inverse d'un critère très contraignant sur la forme des règles, mais n'impliquant pas de limitation technique sur les grammaires. Dans les deux cas, l'apprenabilité découle de l'élasticité finie de ces classes de grammaires, à partir de structures mais également à partir de chaînes (sous réserve que les règles soient strictement décroissantes). Mais nous avons aussi donné un contre-exemple à l'élasticité finie (exemple 6.3) pour les grammaires par consommation d'arguments : celui-ci montre que la frontière de l'élasticité finie est assez vite atteinte, et par conséquent que le cas des grammaires par consommation stricte d'arguments (auxquelles appartiennent les grammaires AB) est relativement isolé.

La problématique de la mise en œuvre de tels algorithmes sur des données réelles a été abordée dans le chapitre 7. Nous y avons étudié un seul aspect du problème, à savoir celui de la nature des données dont l'algorithme d'apprentissage dispose. Afin de permettre que ces données ne soient que de simples phrases tout en limitant l'explosion combinatoire due à l'absence de structures, nous avons soumis l'idée de l'apprentissage partiel. Il s'avère que les grammaires lexicalisées se prêtent relativement bien à cette approche. De plus, la connaissance préalable d'une partie de la grammaire ne semble pas trop contraignante, grâce aux effets de la loi de ZIPF sur les textes en langage naturel. Nous avons utilisé le lexique anglais des grammaires de liens, réalisé par SLEATOR et TEMPERLEY, pour tester cette approche. Les résultats obtenus tendent à montrer la validité de la méthode employée, même si cette première expérimentation est loin d'être exhaustive, en particulier en ce qui concerne la quantité et la complexité des données utilisées.

Au cours de cette étude, nous avons vu les nombreuses difficultés que pose la problématique de l'apprentissage automatique de grammaires dans la perspective d'applications aux langues naturelles. Pour résumer, ces difficultés sont d'abord les limites théoriques liées au modèle de GOLD lui-même, mais aussi les problèmes classiques d'efficacité des algorithmes selon les données dont ils disposent. Nous avons apporté dans la présente étude quelques résultats qui contribuent à pallier ces difficultés. Toutefois, les applications concrètes de l'apprentissage de grammaires aux langues naturelles demeurent très éloignées. Sur le plan théorique, le modèle de GOLD est tout de même assez restrictif, même s'il l'est moins qu'on ne le croyait lorsqu'il a été défini. Sur le plan pratique surtout, de nombreux problèmes ne sont pas encore résolus : la question du choix de la grammaire minimale parmi un ensemble de solutions, celle de la complexité algorithmique décuplée lorsqu'on envisage de quitter le cas des grammaires rigides, et enfin celle du passage à des données plus complexes que celles utilisée ici. Le modèle de

GOLD peut-il servir de base à de véritables applications de l'apprentissage sur les langues naturelles ? Plus généralement, l'apprentissage complet de grammaires du langage naturel par la machine est-il possible ? Dans cette étude, nous avons apporté quelques modestes éléments de réponse. Mais il reste bien entendu de nombreux problèmes à résoudre avant de pouvoir répondre de façon satisfaisante à d'aussi vastes questions.

Bibliographie

[1] S. ABRAMSKY, D. M. GABBAY et T. S. E. MAIBAUM, réds. *Handbook of Logic in Computer Science, Vol. I.* Clarendon Press, Oxford, 1992.

[2] M. AIZERMAN, E. BRAVERMAN et L. ROZONOER. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821 – 837, 1964.

[3] Kasimir AJDUKIEWICZ. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935.

[4] D. ANGLUIN. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[5] Dana ANGLUIN. Inductive inference of formal languages from positive data. *Information and Control*, 48:117–135, 1980.

[6] Dana ANGLUIN. Queries and concept learning. *Machine Learning*, 2:319, 1987.

[7] Dana ANGLUIN. Queries revisited. Dans *Algorithmic Learning Theory, 12th International Conference, ALT 2001, Washington, DC, USA, November 25–28, 2001, Proceedings*, volume 2225 de *Lecture Notes in Artificial Intelligence*, pages 12–31. Springer, 2001.

[8] Dana ANGLUIN et Carl H. SMITH. Inductive inference: Theory and methods. *ACM Computing Surveys*, 15(3):237–269, septembre 1983.

[9] Krzysztof R. APT. Logic programming. Dans J. van LEEWEN, réd., *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapitre 10, pages 493–574. The MIT Press, New York, NY, 1990.

[10] S. ARIKAWA, T. SHINOHARA et A. YAMAMOTO. Learning elementary formal systems. *Theoretical Computer Science*, 95(1):97–113, mars 1992.

[11] Jean-Michel AUTEBERT. *Théorie des langages et des automates*. Masson, 1994.

[12] Y. BAR-HILLEL, C. GAIFMAN et E. SHAMIR. On categorial and phrase structure grammars. *Bulletin of the Research Council of Israel*, 9F:1–16, 1960.

[13] Yehoshua BAR-HILLEL. A quasi-arithmetical notation for syntactic description. *Language*, 29, 1953.

[14] Denis BÉCHET. k-valued link grammars are learnable from strings. Dans *Proceedings of Formal Grammars 2003*, pages 9–18, 2003.

[15] Denis BÉCHET, Alexander DIKOVSKY, Annie FORET et Erwan MOREAU. On learning discontinuous dependencies from positive data. Dans G. JÄGER, P. MONACHESI, G. PENN et S. WINTER, réds., *Proceedings of the 9th conference on Formal Grammar (FG 2004)*, pages 1–16, Nancy, France, août 2004.

[16] Denis BÉCHET et Annie FORET. Apprentissage des grammaires de lambek rigides et d'arité bornée pour le traitement automatique des langues. Dans *Actes de la Conférence d'APprentissage 2003 (CAP'2003)*, 2003.

[17] Jérôme BESOMBES et Jean-Yves MARION. Identification of reversible dependency tree languages. Dans Luboš POPELÍNSKÝ et Miloslav NEPIL, réds., *Proceedings of the 3d Workshop on Learning Language in Logic*, pages 11–22, Strasbourg, France, septembre 2001.

[18] Jérôme BESOMBES et Jean-Yves MARION. Learning reversible categorial grammars from structures. Dans *Proceedings of Categorial Grammars 2004, Montpellier, France*, pages 148–163, juin 2004.

[19] Jérôme BESOMBES et Jean-Yves MARION. Apprentissage de langages réguliers d'arbres et applications. *Traitement Automatique des Langues*, 44(1):121–153, 2003.

[20] M. BLUM et L. BLUM. Towards a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.

[21] Roberto BONATO. A study on learnability of rigid lambek grammars. Tesi di laurea & mémoire de dea, Universita di Verona & Université Rennes 1, 2000.

[22] Roberto BONATO. *An Integrated Computational Approach to Binding Theory*. Thèse de Doctorat, Università degli Studi di Verona & Université Bordeaux 1, 2006.

[23] Roberto BONATO et Christian RETORÉ. Learning rigid lambek grammars and minimalist grammars from structured sentences. Dans Luboš POPELÍNSKÝ et Miloslav NEPIL, réds., *Proceedings of the 3d Workshop on Learning Language in Logic*, pages 23–34, Strasbourg, France, September 2001.

[24] Joan BRESNAN. *Lexical-Functional Syntax*. Blackwell, Oxford, 2001.

[25] Eric BRILL. *A Corpus-Based Approach to Language Learning*. Thèse de Doctorat, Computer and Information Science, University of Pennsylvania, juin 1993.

[26] Wojciech BUSZKOWSKI. Mathematical linguistics and proof theory. Dans Johan VAN BENTHEM et Alice TER MEULEN, réds., *Handbook of Logic and Language*, pages 683–736. Elsevier, Amsterdam, 1996.

[27] Wojciech BUSZKOWSKI et Gerald PENN. Categorial grammars determined from linguistic data by unification. Rapport technique TR-89-05, Department of Computer Science, University of Chicago, juin 1989.

[28] Claudia CASADIO. Semantic categories and the development of categorial grammars. Dans E. Bach R. OEHRLE et D. WHEELER, réds., *Categorial Grammars and Natural Language Structures*, pages 95–124. Reidel, Dordrecht, 1988.

[29] Noam CHOMSKY. On certain formal properties of grammars. *Info. and Control*, 2(2):137–167, 1959.

[30] Stephen CLARK, Julia HOCKENMAIER et Mark STEEDMAN. Building deep dependency structures using a wide-coverage CCG parser. Dans *ACL*, pages 327–334, 2002.

[31] J. COCKE et J. T. SCHWARTZ. Programming languages and their compilers : Preliminary notes. Rapport technique, New York University, 1970.

[32] Christophe COSTA FLORÊNCIO. Combinatory categorial grammars and finite elasticity. Dans Véronique HOSTE et Guy De PAUW, réds., *Proceedings of the Eleventh Belgian-Dutch Conference on Machine Learning*, pages 13–18. University of Antwerp, 2001.

[33] Christophe COSTA FLORÊNCIO. Consistent Identification in the Limit of the Class k-valued is NP-hard. Dans Philippe de GROOTE, Glyn MORRILL et Christian RETORÉ, réds., *Logical Aspects of Computational Linguistics, 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings*, volume 2099 de *Lecture Notes in Computer Science*, pages 125–138. Springer-Verlag, 2001.

[34] Christophe COSTA FLORÊNCIO. Consistent Identification in the Limit of Rigid Grammars from Strings is NP-hard. Dans P. ADRIAANS, H. FERNAU et M. van ZAANEN, réds., *Grammatical Inference: Algorithms and Applications 6th International Colloquium: ICGI 2002*, volume 2484 de *Lecture Notes in Artificial Intelligence*, pages 49–62. Springer-Verlag, September 23-25 2002.

[35] Christophe COSTA FLORÊNCIO. *Learning categorial grammars*. Thèse de Doctorat, Utrecht University, 2003.

[36] Elias DAHLHAUS et Manfred K. WARMUTH. Membership for growing context-sensitive grammars is polynomial. *JCSS: Journal of Computer and System Sciences*, 33:456–472, 1986.

[37] Dick de JONGH et Makoto KANAZAWA. Angluin's theorem for indexed families of r.e. sets and applications. Dans *COLT '96: Proceedings of the ninth annual conference on Computational learning theory*, pages 193–204. ACM Press, 1996.

[38] Colin de la HIGUERA. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.

[39] Colin de la HIGUERA. A bibliographical study of grammatical inference. *Pattern Recognition*, 2005.

[40] M. DEKHTYAR et A. DIKOVSKY. Categorial dependency grammars. Dans M. MOORTGAT, réd., *Proceedings of Categorial Grammars 2004*, pages 76–91, Montpellier, France, juin 2004.

[41] Alexandre DIKOVSKY. Dependencies as categories. Dans G-J. KRUIFF et D. DUCHIER, réds., *Proceedings of Workshop "Recent Advances in Dependency Grammars"*, Geneva, Switzerland.

[42] D. DUDAU-SOFRONIE et I. TELLIER. Un modèle d'acquisition de la syntaxe à l'aide d'informations sémantiques. Dans *actes de la 11ème Conférence TALN, Traitement Automatique du Langage Naturel*, pages 137–146, Fès, Maroc, avril 2004.

[43] Daniela DUDAU-SOFRONIE. *Apprentissage de grammaires catégorielles pour simuler l'acquisition du langage naturel à l'aide d'informations sémantiques*. Thèse de Doctorat, Université Lille 1, avril 2004.

[44] Pierre DUPONT et Laurent MICLET. Inférence grammaticale régulière : fondements théoriques et principaux algorithmes. Rapport technique PI-1189, IRISA, juillet 1998.

[45] J. EARLEY. An efficient context-free parsing algorithm. *Communication of the Association for Computing Machinery*, 13(2):94–102, 1970.

[46] Annie FORET et Yannick LE NIR. Lambek rigid grammars are not learnable from strings. Dans *COLING'2002, 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

[47] Annie FORET et Yannick LE NIR. On limit points for some variants of rigid lambek grammars. Dans *ICGI'2002, the 6th International Colloquium on Grammatical Inference*, number 2484 dans Lecture Notes in Artificial Intelligence. Springer-Verlag, 2002.

[48] Mark FULK, Sanjay JAIN et Daniel OSHERSON. Open problems in systems that learn. *Journal of Computer and System Sciences*, 49(3):589–604, 1994.

[49] Jean-Yves GIRARD. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[50] E.M. GOLD. Language identification in the limit. *Information and control*, 10(5):447–474, March 1967.

[51] S. A. GREIBACH. A new normal form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12:42–52, 1965.

[52] Dennis GRINBERG, John LAFFERTY et Daniel SLEATOR. A robust parsing algorithm for LINK grammars. Rapport technique CMU-CS-TR-95-125, Carnegie Mellon University, Pittsburgh, PA, 1995.

[53] M. HEPPLE. *The Grammar and Processing of Order and Dependency: A Categorial Approach*. Thèse de Doctorat, Centre for Cognitive Science, University of Edinburgh., 1990.

[54] James HIGGINBOTHAM. English is not a context-free language. *Linguistic Inquiry*, 15(2):225–234, 1984.

[55] Julia HOCKENMAIER. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Thèse de Doctorat, School of Informatics, The University of Edinburgh, 2003.

[56] J.E. HOPCROFT et J.D. ULLMAN. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[57] Kent JOHNSON. Gold's theorem and cognitive science. *Philosophy of Science*, 71:571–592, 2004.

[58] A. JOSHI et Y. SCHABES. Tree-adjoining grammars, 1997.

[59] Makoto KANAZAWA. A note on language classes with finite elasticity. Rapport technique CS-R9471, Centrum voor Wiskunde en Informatica (CWI), décembre 1994.

[60] Makoto KANAZAWA. *Learnable classes of categorial grammars*. Cambridge University Press, 1998.

[61] Shyam KAPUR. *Computational Learning of Languages*. Thèse de Doctorat, Department of Computer Science, Cornell University, Ithaca, New York, 1991.

[62] T. KASAMI. An efficient recognition and syntax analysis algorithm for context-free languages. Rapport technique AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts, 1965.

[63] Kevin KNIGHT. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1):93–124, mars 1989.

[64] Joachim LAMBEK. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[65] Gary MARCUS. Negative evidence in language acquisition. *Cognition*, 46:53–85, 1993.

[66] Igor MEL'CUK. Dependency in linguistic description.

[67] Richard MONTAGUE. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, 1974. Edited and with an introduction by Richard Thomason.

[68] Edward F. MOORE. Gedanken-experiments on sequential machines. Dans C. E. SHANNON et J. MCCARTHY, réds., *Automata Studies*, pages 129–153, Princeton, 1956. Princeton University Press.

[69] Michael MOORTGAT. *Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus*. Foris, Dordrecht, 1988.

[70] Michael MOORTGAT. Categorial type logics. Dans Johan VAN BENTHEM et Alice TER MEULEN, réds., *Handbook of Logic and Language*, pages 93–177. Elsevier, Amsterdam, 1997.

[71] Michael MOORTGAT et Richard MOOT. Cgn to grail: Extracting a type-logical lexicon from the cgn annotation. Dans *Proceedings of CLIN 2000*. W. Daelemans, 2001.

[72] Richard MOOT. A short introduction to grail. Dans C. ARECES et M. de RIJKE, réds., *Proceedings of Methods for Modalities 2*, 2001.

[73] Richard MOOT. Parsing corpus-induced type-logical grammars. Dans R. BERNARDI et M. MOORTGAT, réds., *Proceedings of the CoLogNet/ElsNet Workshop on Linguistic Corpora and Logic Based Grammar Formalisms*, 2003.

[74] Erwan MOREAU. Apprentissage des gramaires catégorielles et de dépendances : grammaires catégorielles avec itérations. Rapport de dea, Université de Nantes, 2001.

[75] Erwan MOREAU. Apprentissage partiel de grammaires lexicalisées. *Traitement Automatique des Langues (TAL)*, 45(3):71–102, 2004.

[76] Erwan MOREAU. From link grammars to categorial grammars. Dans M. MOORTGAT, réd., *Proceedings of Categorial Grammars 2004*, pages 31–45, Montpellier, France, juin 2004.

[77] Erwan MOREAU. Partial learning using link grammars data. Dans G. PALIOURAS et Y. SAKAKIBARA, réds., *Grammatical Inference: Algorithms and applications. 7th International Colloquium: ICGI 2004*, volume 3264 de *Lectures Notes in Artificial Intelligence*, pages 211–222, Athens, Greece, October 2004. Springer.

[78] Erwan MOREAU. Learnable classes of general combinatory grammars. Dans J. Busquets P. BLACHE, E. STABLER et R. MOOT, réds., *5th International Conference, LACL 2005*, volume 3492 de *Lectures Notes in Artificial Intelligence*, pages 189–204, Bordeaux, France, April 2005. Springer.

[79] Glyn MORRILL. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Publisher, Dordrecht, 1994.

[80] T. MOTOKI, T. SHINOHARA et K. WRIGHT. The correct definition of finite elasticity: corrigendum to Identification of unions. Dans *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, page 375, San Mateo, CA, 1991. Morgan Kaufmann.

[81] J. NICOLAS. Grammatical inference as unification. Rapport technique 3632, INRIA, juillet 1999. également rapport IRISA PI1265.

[82] Yannick Le NIR. *Structures des analyses syntaxiques catégorielles. Application à l'inférence grammaticale*. Thèse de Doctorat, Université de Rennes 1, 2003.

[83] Daniel N. OSHERSON, Michael STOB et Scott WEINSTEIN. *Systems That Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, Cambridge, Massachusetts, 1986.

[84] R. PAREKH et V. HONAVAR. Learning DFA from simple examples. Dans Ming LI et Akira MARUOKA, réds., *Proceedings of the 8th International Workshop on Algorithmic Learning Theory (ALT-97)*, volume 1316 de *LNAI*, pages 116–131, Berlin, octobre 6–8 1997. Springer.

[85] M. PENTUS. Lambek calculus and formal grammars. Dans *Provability, Complexity, Grammars*, number 192 dans American Mathematical Society Translations–Series 2, pages 57–86. American Mathematical Society, Providence, Rhode Island, 1999.

[86] Mati PENTUS. Lambek calculus is np-complete. Rapport technique TR-2003005, CUNY Ph.D. Program in Computer Science, 2003.

[87] Guy PERRIER et Bertrand GAIFFE. Traitement automatique des langues - concepts et algorithmes, 2004. Cours DEA Informatique PRTAL 3.

[88] S. PINKER. *The Language Instinct: How the Mind Creates Language*. HarperCollins, New York, 1994.

[89] Carl POLLARD et Ivan SAG. *Head-Driven Phrase Structure Grammar*. Chicago University Press, Chicago, Illinois, 1994.

[90] Christian RETORÉ. The logic of categorial grammars. pages 11–14, Birmingham, 2000. Lecture notes ESSLLI'2000.

[91] Christian RETORÉ et Edward STABLER. Resource logics and minimalist grammars. Technical-report, Inria, Institut National de Recherche en Informatique et en Automatique, 1999.

[92] J. A. ROBINSON. A machine oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965.

[93] Yasubumi SAKAKIBARA. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45, octobre 1997.

[94] Takeshi SHINOHARA. Inductive inference from positive data is powerful. Dans *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 97–110. Morgan Kaufmann, 1990.

[95] Takeshi SHINOHARA. Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, 8(4):371–384, 1991.

[96] Takeshi SHINOHARA. Rich classes inferrable from positive data: Length-bounded elementary formal systems. *Information and Computation*, 108:175, February 1994.

[97] Klaas SIKKEL et Anton NIJHOLT. Parsing of context-free languages. Dans A. Salomaa G. ROZENBERG, réd., *Handbook of Formal Languages*, volume 2: Linear Modeling: Background and Application, pages 61–100. Springer Verlag, Berlin, 1997.

[98] Daniel D. K. SLEATOR et Davy TEMPERLEY. Parsing english with a link grammar. Rapport technique CMU-CS-TR-91-126, Carnegie Mellon University, Pittsburgh, PA, 1991.

[99] R. SMULLYAN. *Theory of Formal Systems*. Princeton University Press, Princeton, New Jersey, 1961.

[100] D. Dudau SOFRONIE, I. TELLIER et M. TOMMASI. A tool for language learning based on categorial grammars and semantic information. Dans P. ADRIAANS, H. FERNAU et M. van ZAANEN, réds., *Grammatical Inference: Algorithms and Applications; 6th International Colloquium, ICGI 2002*, volume 2484 de *LNCS/LNAI*, pages 303–305. Springer, 2002.

[101] Edward STABLER. Derivational minimalism. Dans Christian RETORÉ, réd., *Logical Aspects of Computational Linguistics*, pages 68–95, Berlin, 1997. Springer. LNAI 1328.

[102] Mark STEEDMAN. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439, 1987.

[103] Mark STEEDMAN. *The Syntactic Process*. The MIT Press, Cambridge, Massachusetts, 2000.

[104] Mark STEEDMAN et Jason BALDRIDGE. Combinatory categorial grammars, 2003. ftp://ftp.cogsci.ed.ac.uk/pub/steedman/ccg/manifesto.pdf.

[105] Isabelle TELLIER. Modéliser l'acquisition de la syntaxe du langage naturel via l'hypothèse de la primauté du sens, 2005. Habilitation à Diriger des Recherches, Université Lille 3.

[106] Davy TEMPERLEY, Daniel SLEATOR et John LAFFERTY. Link grammar. http://hyper.link.cs.cmu.edu/link/, 1991.

[107] Lucien TESNIERE. *Elements de syntaxe structurale*. Klincksieck, Paris, 1959.

[108] L. G. VALIANT. A theory of the learnable. Dans *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445, New York, NY, USA, 1984. ACM Press.

[109] K. WRIGHT. Identification of unions of languages drawn from an identifiable class. Dans *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 328–333. Morgan Kaufmann, 1989.

[110] Daniel H. YOUNGER. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, février 1967.

Liste des algorithmes

Liste des figures

Table des matières

Index alphabétique

Acquisition de grammaires lexicalisées pour les langues naturelles

Erwan MOREAU

Résumé

L'inférence grammaticale désigne le problème qui consiste à découvrir les règles de formation des phrases d'un langage, c'est-à-dire une grammaire de celui-ci. Dans le modèle d'apprentissage de Gold, les exemples fournis sont constitués uniquement des phrases appartenant au langage. L'algorithme doit fournir une grammaire qui représente le langage énuméré. Les grammaires catégorielles sont l'un des nombreux formalismes existants pour représenter des langages. Kanazawa a montré que certaines sous-classes de ces grammaires sont apprenables, mais ses résultats ne sont pas applicables directement aux langues naturelles.

Sur le plan théorique, nous proposons de généraliser les résultats de Kanazawa à différents types de grammaires. Les grammaires combinatoires générales sont un modèle flexible permettant de définir des systèmes grammaticaux à base de règles de réécriture. Nous démontrons dans ce cadre que certaines classes de langages sont apprenables. Dans un souci de généralité maximale, nos résultats sont exprimés sous forme de critères sur les règles des systèmes grammaticaux considérés. Ces résultats sont appliqués à plusieurs formalismes relativement adaptés à la représentation des langues naturelles.

Nous abordons également le problème de la mise en œuvre de l'apprentissage sur des données réelles. En effet, les algorithmes existants capables d'apprendre des classes de langages intéressantes sont NP-complets. Afin de contourner cet obstacle, nous proposons un cadre d'apprentissage plus souple, l'apprentissage partiel : le contexte d'utilisation est modifié dans le but d'obtenir une complexité algorithmique plus réaliste. Nous testons cette approche sur des données de taille moyenne, et obtenons des résultats relativement encourageants.

Mots-clés : Apprentissage automatique, Inférence grammaticale, Modèle de Gold, Identification à la limite, Grammaires lexicalisées, Grammaires catégorielles, Langues naturelles.

Abstract

Grammatical inference consists in discovering the rules governing how sentences of a language are formed, that is a grammar of this language. In Gold's model of learning, examples given as input are only sentences belonging to the language. The algorithm must provide a grammar which represents the enumerated language. Categorial grammars are one of the numerous existing formalisms used to represent languages. Kanazawa has shown that some subclasses of these grammars are learnable, but his results do not apply directly to natural languages.

In a theoretical viewpoint, we propose to generalize Kanazawa's results to different kinds of grammars. General combinatory grammars are a flexible model that permits to define grammatical systems based on rewriting rules. In this framework, we show that some classes of languages are learnable. In order to be maximally general, our results are expressed in the form of criteria on the grammatical system rules. These results are applied to several formalisms which are quite well suited for the representation of natural languages.

We also address the problem of implementing learning algorithms with real data. Indeed, existing algorithms that are able to learn rich classes of languages are NP-complete. We propose a more flexible learning framework, called partial learning, to bypass this obstacle: the context in which learning takes place is modified, in order to obtain a more realistic algorithmic complexity. We test this approach with some average size data, and obtain quite encouraging results.

Keywords: Automatic learning, Grammatical inference, Gold's model, Identification in the limit, Lexicalized grammars, Categorial grammars, Natural languages.