Infering Weighted Abstract Categorial Grammars

Mémoire de fin d'étude

Master *Sciences et Technologies*,
Mention *Informatique*,
Parcours IASD

Auteur
Dylan Bettendroffer

Superviseurs
Philippe de Groote
Sylvain Pogodalla

Lieu de stage
LORIA - Campus Scientifique - BP 239 - 54506 Vandoeuvre-lès-Nancy

soutenu publiquement le 10 juin 2024

Résumé

Ce mémoire de master traite de l'encodage de formalismes de grammaires probabilistes (HMM, pCFG, pTAG) dans les grammaires catégorielles abstraites.

Après avoir défini les grammaires catégorielles abstraites, les formalismes de grammaires que nous allons encoder, et les grammaires catégorielles abstraites multitypées et pondérées, nous décrivons différents encodages avant de montrer qu'ils sont équivalents aux formalismes de grammaire qu'ils encodent.

Abstract

This master's thesis deals with the encoding of stochastic grammar formalisms (HMM, pCFG, pTAG) in abstract categorial grammars.

After defining those grammar formalisms, abstract categorial grammars, and the weighted and multityped variants of abstract categorial grammars, we describe how to encode said formalisms before showing that they are indeed equivalent in terms of expressiveness to their encoding.

Table des matières

CHAPITRE 1

Introduction

1.1 Contexte

Les grammaires catégorielles abstraites (ACG, de Groote, 2001) sont un cadre grammatical permettant l'encodage de différents formalismes de grammaire. Elles font partie de la famille des grammaires catégorielles (grammaires AB (Bar-Hillel, 1953), grammaires de Lambek (Lambek, 1958 , Moot & Retoré, 2012)) et présentent un ensemble de propriétés intéressantes d'un point de vue théorique et computationnel comme la possibilité de les composer et leur réversibilité. Cette propriété est par exemple utile lorsque l'on veut générer du texte plutôt que d'en analyser. Les grammaires non contextuelles (CFG, de Groote & Pogodalla, 2004) et les grammaires d'arbres adjoints (TAG, Pogodalla, 2017) ont déjà été encodées dans les ACG.

Les ACG peuvent cependant souffrir de problèmes similaires à ceux rencontrés par les grammaires qu'elles encodent. Sur des grammaires de grandes tailles, une explosion combinatoire des ambiguïtés d'analyse pour un même texte peut se produire. Classiquement, la manière de limiter cette explosion est l'utilisation de règles pondérées où les pondérations représentent les probabilités d'utiliser cette règle lors d'une dérivation. Les grammaires régulières ont comme variante probabiliste les modèles de Markov cachés (HMM, Manning & Schütze, 1999), les grammaires non contextuelles et les grammaires d'arbres adjoints ont toutes deux des variantes probabilistes (resp. pCFG (Manning & Schütze, 1999) et pTAG (Resnik, 1992)).

Ces grammaires probabilistes présentent également un avantage majeur : leurs poids peuvent être inférés de corpus de textes (Chiang, 2000) ; voire parfois, la grammaire complète peut s'inférer de corpus annotés. Il existe dans la littérature à ce sujet des algorithmes comme l'algorithme forward-backward pour les HMM ou l'algorithme inside-outside pour les pCFG.

1.2 Problématique

Il est souhaitable en traitement automatique des langues et en linguistique computationnelle de conserver un historique des dérivations utilisées lors de l'analyse d'un texte. Les

grammaires formelles permettent de conserver cette trace ainsi que de renseigner, dans le cas des grammaires probabilistes, sur la probabilité d'occurence d'une chaine dans notre langage. L'émergence des grands modèles de langue pourrait laisser penser que ces objets d'études sont devenus obsolètes, mais ces bénéfices ne s'y retrouvent pas. Ces derniers se comportent comme des boites noires, et cette limitation est inhérente à leur conception.

Les travaux de Ludmann, Pogodalla, et de Groote (2022) ont amené au développement d'ACG pondérés (wACG), que nous détaillons plus loin dans ce rapport. Ces travaux se sont intéressés à l'encodage des formalismes probabilistes susnommés dans les wACG.

La composition des ACG est une opération permise par leur structure. Les ACG sont composés de deux vocabulaires et d'une fonction de lexique. Ainsi, si nous avons deux ACG dont le premier a pour vocabulaire cible le vocabulaire formant le domaine du lexique du second, il nous suffit de composer ces lexiques pour obtenir leur composition. Cette opération existe aussi et est bien définie pour les wACG, même si elle est un peu plus complexe.

Nous traitons ici de l'encodage dans ces wACG de grammaires stochastiques. Nous commencerons par détailler les pré-requis nécessaires à l'encodage de grammaires dans les ACG.

Ensuite, nous décrirons les ACG en tant que cadre pour l'encodage de grammaires ainsi que leurs variantes pondérées et multitypées.

Et enfin, nous définirons les différents encodages de grammaires dans les ACG avant de montrer la correction (au sens d'équivalence) de ces encodages.

CHAPITRE 2

Pré-requis

Les résultats de ce mémoire portant sur l'encodage de grammaires pondérées dans les grammaires catégorielles abstraites, nous définissons dans cette partie les notions que nous mobiliserons dans les chapitres 6, 7 et 8. Chaque grammaire est accompagnée de sa variante stochastique ainsi que de notions nécessaires aux preuves d'équivalences entre les grammaires et leur encodage.

Il est courant dans le domaine des grammaires formelles de différencier les structures manipulées par la grammaire de celles qu'elle produit. Pour des raisons qui paraîtront plus évidentes plus tard, nous parlerons ici de structures abstraites pour les premières et de structures objets pour les secondes. Dans le cas du traitement automatique du langage naturel, les structures objets d'une grammaire sont généralement des chaînes de caractères (ou des chaînes de mots) mais les structures abstraites peuvent fortement varier d'un formalisme à un autre. Nous préciserons donc également quelles sont les structures abstraites et les structures objets de chaque formalisme.

2.1 Grammaires régulières

Automates d'états finis et langages réguliers

Définition 1 (Automate d'état fini, Jurafsky & Martin, 2000)**.** *Un automate est un quintuplet* $(\Sigma, S, E, F, \delta)$ *où :*

- Σ *est un alphabet ;*
- S *est un ensemble fini d'états ;*
- E *est l'ensemble des états initiaux de l'automate ;*
- F *est l'ensemble des états acceptants de l'automate ;*
- $\delta : S \times \Sigma \to S$ *est une fonction de transition. Autrement dit, une fonction qui a un état* $q \in S$ *et à un symbole* $i \in \Sigma$ *va associer un nouvel état* $q' \in S$*.*

Si F est un singleton, on parle d'automate d'état fini déterministe (DFA). Si F est un sous-ensemble de S ou que δ est une fonction de $S \times \Sigma$ dans l'ensemble des parties de S on parle d'automate d'état fini non déterministe (NFA).

Les langages pouvant être reconnus par des automates d'état finis sont appelés **langages réguliers** et on parle de **grammaires régulières** pour les grammaires produisant ces langages.

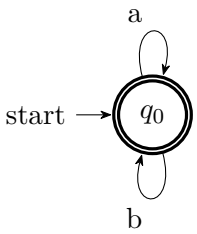

FIGURE 2.1 – Automate reconnaissant la grammaire $(a \mid b)^*$

L'automate de la figure 2.1 reconnaît le langage régulier $(a \mid b)^*$ avec $*$ l'étoile de Kleene. Son unique état q_0 est état entrant et état acceptant, et il reconnaît toute suite de a et de b de taille arbitraire.

Ici les structures abstraites de la grammaire sont les marches dans le graphe que représente l'automate, généralement représentées par un couple $(O, X) \in \Sigma^* \times Q^*$ où $O = o_1 + \cdots + o_k$ est la chaîne de symboles de Σ produite par la suite d'état $X = q_1 q_2 \dots q_k$. Nous soulignerons que par notre définition de δ la taille de O est égale à la taille de X.

Modèle de Markov cachés (HMM)

Les automates d'états finis ont plusieurs variantes probabilistes qui ajoutent généralement des poids sur les transitions. Les modèles de Markov cachés diffèrent des automates probabilistes classiques en ceci qu'ils sont munis d'états dits "cachés". Ceux-ci prennent la forme de probabilités d'émission de symboles terminaux lors de l'arrivée dans un état.

Nous définissons ensuite un modèle de Markov caché.

Définition 2 (HMM, Manning & Schütze, 1999, Jurafsky & Martin, 2000)**.** *Un modèle de Markov caché est un quintuplet $\mu = (Q, V, A, B, \Pi)$ où :*

- $Q = \{s_1, \dots, s_N\}$ *est un ensemble fini d'états ;*
- $V = \{k_1, \dots, k_M\}$ *est un alphabet fini ;*
- $A = (a_{qq'})_{q,q' \in Q}$ *est la matrice de probabilités de transitions du modèle ; $a_{qq'}$ est la probabilité $P(q', \mid q, \mu)$ de transitionner dans q' depuis q dans μ ;*
- $B = (b_{qv})_{q \in Q, v \in V}$ *est la matrice de probabilité d'émissions du modèle ; b_{qv} est la probabilité $P(v \mid q, \mu)$ d'émettre v en arrivant à l'état q dans μ ;*
- $\Pi = (\pi_q)_{q \in Q}$ *est un vecteur de probabilité des états initiaux du modèle.*

On peut voir que les transitions n'émettent (ou ne reconnaissent) plus de symboles. Mais les HMM doivent respecter un certain nombre de contraintes sur les probabilités de transitions, les probabilités d'entrées et les probabilités d'émissions.

Les matrices A, B et Π sont contraintes comme suit :

$$\forall q \in Q, \sum_{\forall q' \in Q} a_{q,q'} = 1;$$

$$\forall q \in A, \sum_{\forall o \in V} b_{q,o} = 1;$$

$$\sum_{\forall q \in Q} \pi_q = 1;$$

Définition 3 (Poids d'une chaîne dans un HMM)**.** *Soit $\mu = \langle Q, V, A, B, \Pi \rangle$ un modèle de Markov caché, $O = (o_1, o_2, \ldots, o_n)$ une chaîne de symboles de V^n et $X = (q_1, q_2, \ldots, q_n)$ une suite d'états de Q^n. Nous définissons les probabilités suivantes :*

$$\begin{aligned} P(O, q \mid init, \mu) &= P(o_1, q_1 \mid init, \mu) \prod_{i=2}^{n} P(o_i, q_i \mid q_i - 1, \mu) \\ &= \pi_{q_1} b_{q_1, o_1} \prod_{i=1}^{n} a_{q_{i-1}, q_i} b_{q_i, o_i} \end{aligned}$$

$$\begin{aligned} P(O, q, \mid q_0, \mu) &= \prod_{i=1}^{n} P(o_i, q_i \mid q_{i-1}, \mu) \\ &= \prod_{i=1}^{n} a_{q_{i-1}, q_i} b_{q_i, o_i} \end{aligned}$$

$$P(O \mid \mu) = \sum_{X \in Q^n} P(O, X \mid init, \mu)$$

Au même titre que les automates, les structures abstraites de la grammaire sont les suites d'états $X = q_0 q_1 \ldots q_n$ mais augmentées des probabilités d'émissions pour chaque état de X .

2.2 Grammaires non contextuelles

Définition 4 (Grammaire non contextuelle (CFG), Hopcroft, Motwani, & Ullman, 2000)**.** *Une **grammaire non contextuelle** est un quadruplet $\langle N, T, P, S \rangle$ où :*

- *N est un ensemble de symboles non-terminaux ;*
- *T est un ensemble de symboles terminaux ;*
- *P est une relation de $N \times (T \cup N)^*$ qui à des symboles non-terminaux associe une suite de terminaux et de non-terminaux. On parle de règles de production ;*
- *$S \in N$ est un symbole non terminal servant de point d'entrée à la grammaire.*

Les règles de productions $(A, \gamma) \in P$ sont généralement représentées avec une flèche séparant la partie gauche de la partie droite de la règle : $A \to \gamma$.

On ne manipulera par la suite que des grammaires en forme normale de Chomsky. Autrement dit, des grammaires dont les règles sont soit de la forme :

$$A \to BC$$

soit de la forme :

$$A \to a$$

où $A, B, C \in N$ et $a \in T$.

Définition 5 (Dérivations Hopcroft et al., 2000)**.** *Soit une CFG $G = \langle N, T, P, S\rangle$. Soit $\alpha A\beta$ une chaîne de terminaux et de non-terminaux avec $A \in N$ et $\alpha, \beta \in (N \cup T)^*$. Soit une règle de production de P de la forme $A \to \gamma$. Alors $\alpha\gamma\beta$ dérive directement de $\alpha A\beta$ et on le note $\alpha A\beta \Rightarrow \alpha\gamma\beta$.*

La clôture réflexive transitive de la relation $\Rightarrow$ se note avec une étoile $\overset{*}{\Rightarrow}$ et $A \overset{*}{\Rightarrow} \gamma$ signifie que γ est dérivé de A en 0 ou plus étapes.

Définition 6 (Langage d'une CFG)**.** *Le langage d'une CFG $G = \langle N, T, P, S\rangle$ est l'ensemble $\mathcal{L}(G) = \{w \in T^* \mid \exists k, S \overset{k}{\Rightarrow} w\}$.*

Grammaires non contextuelles probabilistes (pCFG)

Définition 7 (Grammaire non contextuelle probabiliste, Manning & Schütze, 1999)**.** *Une grammaire non contextuelle probabiliste est un quintuplet $G = \langle N, T, P, S, \mu\rangle$ où :*

- *$\langle N, T, P, S\rangle$ est une CFG ;*
- *$\mu : P \to [0; 1]$ est une fonction des règles de la grammaire dans l'intervalle des probabilités telle que*

$$\forall i \sum_j A_i \to \alpha_j = 1.$$

Où $A \in N$ est un non-terminal et $\alpha \in (N \cup T)^*$ est une suite de terminaux et non-terminaux.

Définition 8 (Poids d'une dérivation dans une pCFG)**.** *Le poids d'une dérivation d constituées de k règles de production de la forme $A \to \alpha$ avec $\alpha \in (N \cup T)^*$ et $A \in N$ est le produit du poids de ces règles :*

$$\mu(d) = \prod_{i=1}^{k} \mu(A_i \to \alpha_i)$$

Définition 9 (Poids d'un mot d'une pCFG)**.** *Le poids d'un mot w d'une grammaire non contextuelle probabiliste $G = \langle N, T, P, S, \mu\rangle$ est la somme des poids des dérivations produisant w :*

$$\mu(w) = \sum_{\{d|yield(d)=w\}} \mu(d)$$

Définition 10 (Équivalence forte de grammaires, Yoshinaga, Miyao, & Tsujii, 2002)**.** *Deux grammaires G_1 et G_2 sont dites en forte équivalence s'il existe une bijection entre les structures abstraites de G_1 et G_2.*

2.3 Grammaires d'arbres adjoints

Les grammaires d'arbres adjoints (TAG) sont un formalisme de grammaires de manipulation d'arbres introduit pour la première fois dans Joshi, Levy, et Takahashi (1975).

Définition 11 (Grammaires d'arbres adjoints, Joshi et al., 1975)**.** *Une TAG est un quintuplet* $\langle N, T, I, A, S\rangle$ *tel que :*

— N *est un ensemble fini de symboles non terminaux ;*
— T *est un ensemble fini de symboles terminaux ;*
— I *est un ensemble fini d'arbres initiaux ;*
— A *est un ensemble fini d'arbres auxiliaires ;*
— $S \in N$ *est un symbole non terminal distingué ;*

Une TAG consiste donc en un ensemble fini d'arbres élémentaires (où les arbres élémentaires sont les arbres de $I \cup A$) dont les nœuds sont étiquetés par des symboles terminaux et non-terminaux. Les nœuds étiquetés de symboles terminaux ne peuvent être que des feuilles.

Les grammaires manipulées par la suite dans ce mémoire sont dites **lexicalisées**. C'est-à-dire que pour tout arbre élémentaire de la grammaire, il y a exactement une feuille de cet arbre qui est un symbole terminal. On appelle ce symbole terminal **ancre**.

Dans les arbres initiaux, nommés α_i dans la figure 2.2, on annote par convention les feuilles étiquetées de symboles non-terminaux d'une flèche ($\downarrow$).

Dans les arbres auxiliaires, nommés β_i dans la figure 2.2, une des feuilles est du même type que la racine de l'arbre et est annoté d'une étoile ($\star$).

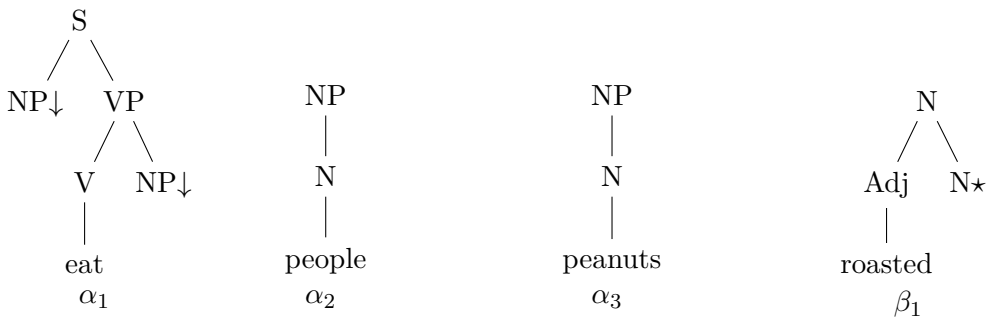

FIGURE 2.2 – Exemple de grammaire d'arbres adjoints

Pour renseigner l'adresse des nœuds d'un arbre, on utilise l'adresse de Gorn.

Définition 12 (Adresse de Gorn, Gorn, 1967)**.** *L'adresse de Gorn d'un nœud dans un arbre enraciné est défini inductivement comme suit :*

— ϵ *si le nœud est la racine de l'arbre ;*
— *sinon si l'indice de Gorn d'un nœud est* i *alors l'indice de son* j*-ème enfant est* $i.j$*.*

Le nœud **VP** l'arbre α_1 dans l'exemple de la figure 2.2 a par exemple pour adresse de Gorn 2. Et le nœud **V** du même arbre a pour adresse 2.1.

Deux opérations élémentaires servent à combiner ces arbres élémentaires pour construire de nouveaux arbres dits dérivés (généralement notés γ) : La **substitution** (illustrée en figure 2.3) et l'**adjonction** (illustrée en figure 2.4).

Définition 13 (Substitution, Joshi et al., 1975)**.** *La substitution d'un arbre γ dans un arbre α à l'adresse de Gorn η de l'arbre α est le remplacement du nœud à l'adresse η par la racine de l'arbre γ. Cette opération n'est permise que si le nœud de α à l'adresse η est une feuille étiquetée d'un non-terminal et que ce non-terminal est le même que celui qui étiquette la racine de γ.*

Seul un arbre initial, ayant possiblement subi des opérations de substitution ou d'adjonctions, peut être substitué à une feuille.

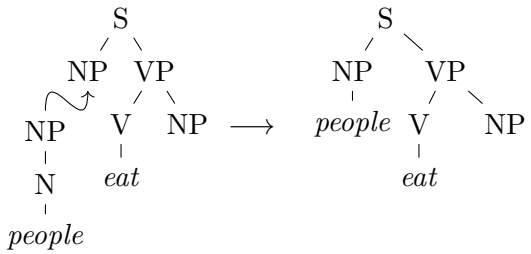

FIGURE 2.3 – Opération de substitution

On peut remarquer que la substitution est très similaire à une opération de réécriture dans une CFG. On peut d'ailleurs voir les CFG comme des TAG non lexicalisées où tous les arbres sont de profondeur 1 (Abeillé, 1993). Mais ceci est hors de la portée de ce mémoire.

Toujours selon Abeillé (1993), L'adjonction est une opération propre aux TAG. Elle insère un arbre auxiliaire (ou dérivé d'un arbre auxiliaire) dans un nœud racine ou un nœud interne d'un arbre élémentaire ou d'un arbre dérivé. Plus formellement :

Définition 14 (Adjonction, Joshi et al., 1975)**.** *L'adjonction est l'opération de suppression d'un nœud interne ou racine d'un arbre élémentaire ou dérivé et la création de plusieurs arêtes, une du père du nœud supprimé vers la racine de l'arbre inséré et une du pied de l'arbre inséré à chacun des enfants du nœud supprimé.*

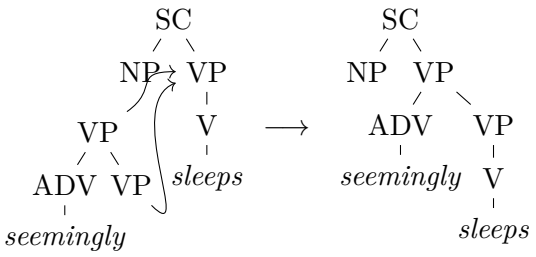

FIGURE 2.4 – Opération d'adjonction

L'historique de ces opérations forme un arbre de dérivations définit comme suit :

Définition 15 (Arbre de dérivation)**.** *Soit γ_1 et γ_2 deux arbres initiaux ou dérivés d'arbres initiaux. L'arbre de dérivation produit par la substitution de γ_1 sur γ_2 à l'adresse de Gorn η est :*

$$\begin{array}{l}\gamma_1\\ \big|\,\eta\\ \gamma_2\end{array}$$

Soient γ un arbre dérivé d'un arbre initial ou un arbre initial et β un arbre dérivé d'un arbre auxiliaire ou un arbre auxiliaire. L'arbre de dérivation produit par l'adjonction de β sur γ à l'adresse de Gorn η est :

$$\begin{array}{l}\gamma\\ \vdots\,\eta\\ \beta\end{array}$$

2.4 Grammaires d'arbre adjoints probabilistes

Les grammaires d'arbres adjoints probabilistes, telles que définies dans (Resnik, 1992), vont encoder trois types de probabilités différentes :

— La probabilité pour un arbre initial d'être la racine d'un arbre de dérivation ;
— la probabilité pour un arbre initial (ou dérivé d'un arbre auxiliaire) d'être substitué dans un autre arbre initial ou dérivé à une adresse donnée ;
— et la probabilité pour un arbre auxiliaire (ou dérivé d'un arbre auxiliaire) d'être adjoint à un arbre initial ou dérivé à une adresse donnée.

Afin de modéliser ces évènements, nous devons définir plusieurs choses. La première est l'ensemble des nœuds d'un arbre α où une substitution est possible que l'on notera $s(\alpha)$, il s'agit de toutes les feuilles étiquetées par un non-terminal et marqués d'une flèche ($\downarrow$). Ensuite, nous définissons l'ensemble des nœuds d'un arbre α où une adjonction est possible, noté $a(\alpha)$, il s'agit ici des nœds interne de l'arbre α qui ne sont pas dans $s(\alpha)$.

Nous noterons ensuite la substitution de α' dans α (α' dans α étant des arbres initiaux) à l'adresse η par $S(\alpha, alpha', \eta)$ avec $\eta ins(\alpha)$.

Puis, pour α un arbre initial et β un arbre auxiliaire, on note l'évènement d'adjonction de β dans α à l'adresse η comme $A(\alpha, \beta, \eta)$, et $A(\alpha, none, \eta)$ l'absence d'adjonction sur le nœud η.

Nous définissons ensuite Ω comme l'union pour tout $\alpha, \alpha' \in I, \beta \in A$ et pour toute adresse de Gorn $\eta \in s(\alpha)$ et $\zeta \in a(\alpha)$ des $S(\alpha, \alpha', \eta)$ et des $A(\alpha, \beta, \zeta)$.

À l'aide de ces ensembles, nous pouvons définir une pTAG comme :

Définition 16 (Grammaire d'arbres adjoints probabiliste)**.** *Une pTAG est un octuplet* $\langle N, T, I, A, S, P_I, P_S, P_A\rangle$ *où :*

- $\langle N, T, I, A, S\rangle$ *est une grammaire TAG ;*
- $P_I : I \to [0;1]$ *associe à chaque arbre initial la probabilité d'être la racine d'un arbre de dérivation et*
$$\sum_{\alpha\in I} P_I(\alpha) = 1;$$
- $P_S : \Omega \to [0;1]$ *associe à chaque évènement de substitution* $S(\alpha, \alpha', \eta)$ *une probabilité d'apparaître dans une dérivation et*
$$\forall\alpha \in I \cup A, \forall\eta \in s(\alpha), \sum_{\alpha'\in I} P_S(S(\alpha, \alpha', \eta)) = 1;$$
- $P_A : \Omega \to [0;1]$ *associe à chaque évènement d'adjonction* $A(\alpha, \beta, \eta)$ *une probabilité d'apparaître dans une dérivation et*
$$\forall\alpha \in I \cup A, \forall\eta \in a(\alpha), \sum_{\beta\in A\cup\{none\}} P_A(A(\alpha, \beta, \eta)) = 1.$$

Le poids d'une dérivation est donc défini comme suit

Définition 17 (Poids d'une dérivation dans une pTAG)**.** *Le poids d'une dérivation* $d = \langle\alpha_0, op_1(\dots), \dots, op_n(\dots)\rangle$ *dans une pTAG* G *avec* $\forall 1 \leq i \leq n, op_i \in \{S, A\}$ *est :*

$$P(d|G) = P_I(\alpha_0)\prod_{i=1}^{n} P_{op_i}(op_i(\dots)).$$

Les chapitres 6, 7 et 8 discutent l'encodage dans des ACG de toutes les grammaires définies dans ce chapitre-ci. Dans les chapitres 3 à 5 nous définissons les ACG et leurs variantes multitypées et pondérées.

CHAPITRE 3

Grammaires catégorielles abstraites

Dans ce chapitre, nous définissons les grammaires catégorielles abstraites ainsi que les prérequis nécessaires à leur compréhension. Ce travail est nécessaire pour l'introduction de notre objet d'étude, les grammaires catégorielles abstraites pondérées et multitypées.

3.1 Grammaires catégorielles abstraites

Cette section décrit le formalisme des ACG ainsi que les règles permettant leur manipulation. La plupart des définitions viennent ou sont inspirées de (de Groote, 2001).

Définition 18 (Types implicatifs linéaire)**.** *Soit A un ensemble de types atomiques. L'ensemble $\mathscr{T}_A$ des types implicatifs construits sur A est défini inductivement comme suit :*

- *si $a \in A$ alors $a \in \mathscr{T}_A$;*
- *si $\alpha, \beta \in \mathscr{T}_A$ alors $(\alpha \multimap \beta) \in \mathscr{T}_A$.*

Nous soulignons également que l'implication linéaire $\multimap$ est associative droite. Ce qui nous permet donc d'écrire pour tous types $a_1, a_2, \ldots, a_n \in \mathscr{T}_A$, $a_1 \multimap a_2 \multimap \cdots \multimap a_n$ pour $a_1 \multimap (a_2 \multimap (\cdots \multimap a_n) \ldots)$.

Définition 19 (Signatures linéaires d'ordre supérieur)**.** *Une signature linéaire d'ordre supérieur Σ est un triplet $\Sigma = \langle A, C, \tau \rangle$ tel que :*

- *A est un ensemble fini de types atomiques ;*
- *C est un ensemble fini de constantes ;*
- *$\tau : C \to \mathscr{T}_A$ est une fonction assignant un type implicatif linéaire de $\mathscr{T}_A$ à chacune des constantes de C.*

Étant donné une signature Σ, on notera A_Σ, C_Σ et τ_Σ chacune de ses composantes.

Définition 20 (λ-termes linéaires)**.** *Soient X un ensemble infini énumérable de variables et Σ une signature linéaire d'ordre supérieur où X et C_Σ sont disjoints. L'ensemble $\Lambda(\Sigma)$ des λ-termes construits sur Σ est défini inductivement comme suit :*

- *pour tout $x \in X$, $x \in \Lambda(\Sigma)$;*
- *pour tout $c \in C$, $c \in \Lambda(\Sigma)$;*
- *pour tout $x \in X$ et pour tout $M \in \Lambda(\Sigma)$, $(\lambda x.M) \in \Lambda(\Sigma)$ si x apparaît exactement une fois dans $\mathcal{FV}(M)$;*
- *pour tous $M, N \in \Lambda(\Sigma)$ tels que les variables libres de M et celles de N sont des ensembles disjoints, $(MN) \in \Lambda(\Sigma)$. L'application étant associative gauche, on notera MNL pour $((MN)L)$.*

Opérations usuelles sur les λ-termes

L'ensemble $\Lambda(\Sigma)$ des λ-termes issus de Σ est muni des opérations usuelles d'α-conversion, β-réduction et η-réduction décrites dans (Barendregt, 1985). Nous les rappelons dans cette section.

Définition 21 (β-redex)**.** *On nomme β-**redex** un λ-terme de la forme $(\lambda x.N)M$.*

Définition 22 (β-contraction)**.** *L'opération de β-**contraction** notée $\rightarrow_{1_\beta}$ est l'application de la règle de réécriture suivante :*

$$(\lambda x.N)M \rightarrow_{1_\beta} N[x := M]$$

Où $N[x := M]$ signifie que l'on substitue de toutes les occurrences de la variable x dans le terme N par le terme M.

Définition 23 (β-réduction)**.** *La β-**réduction** est la clôture réflexive et transitive de la β-contraction et nous la notons $\rightarrow_\beta$.*

Définition 24 (η-contraction)**.** *Soit un terme de la forme $\lambda x.Mx$. L'opération de η-**contraction** notée $\rightarrow_\eta$ est l'application de la règle de réécriture :*

$$\lambda x.Mx \rightarrow_\eta M$$

Définition 25 (α-conversion)**.** *L'α-**conversion** est un renommage de variables liées au sein d'un λ-terme pour éviter, lors de l'application d'un λ-terme à un autre, qu'une variable qui était libre se retrouve liée.*

Définition 26 (β-équivalence)**.** *Il s'agit de la plus petite relation d'équivalence (notée $=_\beta$) contenant $\rightarrow_\beta$ et compatible avec l'application et l'abstraction. Deux λ-termes M et N sont β-équivalents s'il existe un λ-terme L tel que $M \rightarrow_\beta L$ et $N \rightarrow_\beta L$.*

Contexte de typage

Soit Σ une signature linéaire d'ordre supérieur. Un contexte de typage Γ est un ensemble fini de paires $x : \alpha$ telles que x est une variable et $\alpha \in \mathscr{T}_{A_\Sigma}$, et tel que pour tout $x : \alpha, y : \beta \in \Gamma$, si $x = y$ alors $\alpha = \beta$. Pour un tel contexte de typage Γ on note $dom(\Gamma)$ l'ensemble des variables apparaissant dans Γ, i.e., $\{x \in X \mid x : \alpha \in \Gamma \text{ pour } \alpha \in \mathscr{T}_{A_\Sigma}\}$.

Un jugement de type $\Gamma \vdash_\Sigma t : \alpha$ assigne le type $\alpha \in \mathscr{T}_{A_\Sigma}$ au terme $t \in \Lambda(\Sigma)$ dans le contexte Γ si $\Gamma \vdash_\Sigma t : \alpha$ est la racine d'un arbre de dérivation obéissant aux règles de la figure 3.1.

$$\frac{}{x : \alpha \vdash_\Sigma x : \alpha}\,(var) \qquad \frac{\Gamma, x : \alpha \vdash_\Sigma t : \beta}{\Gamma \vdash_\Sigma (\lambda x.t) : \alpha \multimap \beta}\,(abs)$$

$$\frac{\Gamma_1 \vdash_\Sigma t : \alpha \multimap \beta \qquad \Gamma_2 \vdash_\Sigma u : \alpha}{\Gamma_1, \Gamma_2 \vdash_\Sigma (tu) : \beta}\,(app)$$

$$\frac{}{\vdash_\Sigma c : \tau_\Sigma(c)}\,(cst) \qquad \text{Si } dom(\Gamma_1) \cap dom(\Gamma_2) = \varnothing$$

FIGURE 3.1 – Règles de typage du λ-calcul linéaire

Morphismes de types et de termes

Définition 27 (Morphismes de types)**.** *Soient A et B deux ensembles de types atomiques. Une application $h : \mathscr{T}_A \to \mathscr{T}_B$ est appelée homomorphisme de types si $\forall \alpha, \beta \in \mathscr{T}_A, h(\alpha \multimap \beta) = h(\alpha) \multimap h(\beta)$.*

Définition 28 (Morphismes de termes)**.** *Soient Σ_1 et Σ_2 deux signatures linéaires abstraites. Une application $h : \Lambda(\Sigma_1) \to \Lambda(\Sigma_2)$ est appelée homomorphisme de termes si $\forall t, u \in \Lambda(\Sigma_1), x \in X, h(x) = x, h(\lambda x.t) = \lambda x.h(t), (tu) = (h(t)(h(u)))$.*

On remarquera que le morphisme de types (resp. de termes) est entièrement défini par la façon dont il envoie les types atomiques de A sur les types linéaires implicatifs de $\mathscr{T}_B$ (resp. les constantes de Σ_1 sur les λ-termes de $\Lambda(\Sigma_2)$).

Définition 29 (Lexique)**.** *Soient deux signatures linéaires abstraites $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ et $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$. Un lexique $\mathscr{L} : \Sigma_1 \to \Sigma_2$ est une paire $\langle F, G \rangle$ telle que :*

- *$F : \mathscr{T}_{A_{\Sigma_1}} \to \mathscr{T}_{A_{\Sigma_2}}$ est un morphisme de types qui interprète les types linéaires implicatifs construits sur Σ_1 comme des types implicatifs linéaires construits sur Σ_2.*
- *$G : \Lambda(\Sigma_1) \to \Lambda(\Sigma_2)$ est un morphisme de termes qui interprète les λ-termes linéaires construits sur Σ_1 comme des λ-termes linéaires construits sur Σ_2 ;*
- *L'interprétation des fonctions est compatible avec la relation de typage, i.e., pour tout $c \in C_1$, $\vdash_{\Sigma_2} G(c) : F(\tau_1(c))$ est dérivable.*

Dans la suite on notera $\mathscr{L}(a)$ pour $F(a)$ ou $G(a)$ si le contexte le permet.

Définition 30 (ACG)**.** *Une grammaire catégorielle abstraite est un quadruplet $\mathscr{G} = \langle \Sigma_{abs}, \Sigma_{obj}, \mathscr{L}, s \rangle$ où :*

- *Σ_{abs} et Σ_{obj} sont deux signatures linéaires d'ordre supérieur. On les nomme respectivement vocabulaire abstrait et vocabulaire objet. Les termes de $\Lambda(\Sigma_{abs})$ (resp. $\Lambda(\Sigma_{obj})$) sont appelés termes abstraits (resp. termes objets).*
- *$\mathscr{L} : \Sigma_{abs} \to \Sigma_{obj}$ est un lexique du vocabulaire abstrait au vocabulaire objet.*
- *$s \in \mathscr{T}_{A_{\Sigma_{abs}}}$ est un type abstrait appelé type distingué de la grammaire.*

Définition 31 (Composition)**.** *Soient deux ACG* $\mathscr{G}_1 = \langle \Sigma_1, \Sigma_2, \mathscr{L}_1, s_1 \rangle$ *et* $\mathscr{G}_2 = \langle \Sigma_2, \Sigma_3, \mathscr{L}_2, s_2 \rangle$*. La composition de ces deux ACG est l'ACG* $\mathscr{G}_2 \circ \mathscr{G}_1 = \langle \Sigma_1, \Sigma_3, \mathscr{L}_2 \circ \mathscr{L}_1, s_1 \rangle$ *où le vocabulaire abstrait de* $\mathscr{G}_2 \circ \mathscr{G}_1$ *est le vocabulaire abstrait de* $\mathscr{G}_1$*, son vocabulaire objet est celui de* $\mathscr{G}_2$ *et sa fonction de lexique est la composition fonctionnelle* $\mathscr{L}_2 \circ \mathscr{L}_1$ *des lexiques des deux grammaires.*

Définition 32 (Langages abstraits et objets)**.** *Soit* $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$ *une ACG. Le* ***langage abstrait*** *de* $\mathscr{G}$ *est :*

$$\mathcal{A}(\mathscr{G}) = \{t \in \Lambda(\Sigma_1) \mid \vdash_{\Sigma_1} t : s\},$$

l'ensemble des λ*-termes du type distingué* s *issus du vocabulaire abstrait. Son* ***langage objet*** *est :*

$$\mathcal{O}(\mathscr{G}) = \{u \in \Lambda(\Sigma_2) \mid \exists t \in \mathcal{A}(\mathscr{G}), \mathscr{L}(t) = u\},$$

l'ensemble des λ*-termes du vocabulaire objet ayants pour antécédant par* $\mathscr{L}$ *un terme* t *du type distingué* s*.*

Dans une ACG $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$, analyser un terme $u \in \Lambda(\Sigma_2)$ revient à trouver $t \in \mathcal{A}(\mathscr{G})$ tel que $\mathscr{L}(t) = u$.

CHAPITRE 4

ACG multitypées (mACG)

Définition 33 (Squelette de type)**.** *Soient A un ensemble de types atomiques et $☠ = \{\Box\}$. Le squelette d'un type $\alpha \in \mathscr{T}_A$ est $skel_A(\alpha)$ où $skel_A : \mathscr{T}_A \to \mathscr{T}_☠$ est un homomorphisme des types atomiques de $\mathscr{T}_A$ vers l'unique type $\Box$.*

Pour un type $\alpha = A_0 \multimap A_1 \multimap \cdots \multimap A_n$ on a donc $skel_A(\alpha) = \Box \multimap \Box \multimap \cdots \multimap \Box$.

Définition 34 (Signatures surchargées)**.** *Une signature surchargée est un triplet $\Pi = \langle A, C, \tau^+ \rangle$ tel que :*

- *A est un ensemble fini de types atomiques ;*
- *C est un ensemble fini de constantes ;*
- *$\tau^+ : C \to \mathcal{P}_f(\mathscr{T}_A)$ est une fonction qui assigne à chaque constante de C un ensemble non vide fini de types implicatifs linéaires de $\mathscr{T}_A$;*
- *Tous les types d'une même constante partagent un squelette de type : $\forall c \in C, \forall \alpha, \beta \in \tau^+(c), skel_A(\alpha) = skel_A(\beta)$.*

On notera par la suite $\mathscr{T}_\Pi$ pour $\mathscr{T}_A$ et $\Lambda(\Pi)$ pour l'ensemble des termes construits sur C.

Les jugements de types sont toujours définis par induction sur les règles de la figure 3.1, mais la règle de typage des constantes est modifiée comme suit :

$$\frac{\alpha \in \tau^+_\Pi(c)}{\vdash_\Pi c : \alpha}\,(cst+)$$

Ainsi, une même variable peut se voir associer plusieurs types.

Définition 35 (Équivalence de squelettes)**.** *Soit Π une signature surchargée. On note $\sim_\Pi$ la plus petite relation d'équivalence telle que :*

- *s'il existe une constante $c \in \Pi$ telle que $\alpha, \beta \in \tau^+(c)$, alors $\alpha \sim_\Pi \beta$;*
- *si $\alpha_0 \multimap \alpha_1 \sim_\Pi \beta_0 \multimap \beta_1$, alors $\alpha_0 \sim_\Pi \beta_0$ et $\alpha_1 \sim_\Pi \beta_1$;*
- *si $\alpha_0 \sim_\Pi \beta_0$ et $\alpha_1 \sim_\Pi \beta_1$, alors $\alpha_0 \multimap \alpha_1 \sim_\Pi \beta_0 \multimap \beta_1$;*

On dit de α et β qu'ils sont **Π-équivalents** si $\alpha \sim_\Pi \beta$.

Définition 36 (Lexique)**.** *Soient $\Pi = \langle A, C, \tau^+ \rangle$ une signature surchargée et Σ une signature simple. Un lexique de Π à Σ est une paire $\mathscr{L}^+ = \langle F, G \rangle$ telle que :*

- *$F : \mathscr{T}_\Pi \to \mathscr{T}_\Sigma$ est un homomorphisme de types qui interprète les types implicatifs linéaires construits sur Π comme des types implicatifs linéaires construits sur Σ ;*
- *$G : \Lambda(\Pi) \to \Lambda(\Sigma)$ est un homomorphisme de termes qui interprète les λ-termes linéaires construits sur Π comme des λ-termes linéaires construits sur Σ ;*
- *L'homomorphisme de types envoie l'équivalence de squelettes sur l'égalité : $\forall a, b \in A, a \sim_\Pi b \Rightarrow F(a) = F(b)$. Autrement dit, si deux types partagent un même squelette, alors ils doivent être envoyés sur le même type objet par le lexique ;*
- *Les interprétations des homomorphismes sont compatibles avec la relation de typage : $\forall c \in C, \forall \alpha \in \tau^+(c), \vdash_\Sigma G(c) : F(\alpha)$.*

Nous définissons maintenant les ACG multitypées ainsi que leur langage abstrait et leur langage objet.

Définition 37 (ACG multitypées)**.** *Une ACG multitypée (mACG) est un quadruplet $\mathscr{G}^+ = \langle \Pi, \Sigma, \mathscr{L}^+, S \rangle$ tel que :*

- *Π est une signature surchargée ;*
- *Σ est une signature simple ;*
- *$\mathscr{L}^+ : \Pi \to \Sigma$ est un lexique de Π dans Σ ;*
- *S est un ensemble de types abstraits de $\mathscr{T}_\Pi$ Π-équivalents.*

Définition 38 (Langages abstraits et objets de mACG)**.** *Soit $\mathscr{G}^+ = \langle \Pi, \Sigma, \mathscr{L}^+, S \rangle$ une mACG.*

Le langage abstrait de $\mathscr{G}^+$ est $\mathcal{A}(\mathscr{G}^+)$, où $\mathcal{A}(\mathscr{G}^+) = \{t \in \Lambda(\Pi) \mid \exists \alpha \in S, \vdash_\Pi t : \alpha\}$.

Le langage objet de $\mathscr{G}^+$ est $\mathcal{O}(\mathscr{G}^+) = \{u \in \Lambda(\Sigma) \mid \exists t \in \mathcal{A}(\mathscr{G}^+), u = \mathscr{L}^+(t)\}$.

On remarquera que la définition est la même pour les langages abstraits et objets d'une mACG que pour une ACG. À ceci près qu'au lieu de chercher les termes d'un type distingué s, on cherche ici les termes qui sont d'un type présent dans un ensemble de types distingués S.

4.1 Composition de mACG

La composition des mACG n'est pas aussi triviale que celle des ACG. En effet, le lexique d'une mACG envoie une signature surchargée sur une signature simple. Ainsi, il ne sera jamais possible de composer deux lexiques $\mathscr{L}_0^+$ et $\mathscr{L}_1^+$ comme $\mathscr{L}_1^+ \circ \mathscr{L}_0^+$ (au sens de composition fonctionnelle classique) car $\mathbf{dom}(\mathscr{L}_1^+) \neq \mathbf{codom}(\mathscr{L}_0^+)$.

Afin de résoudre ce problème, il nous faut nous ramener à des structures connues. C'est pourquoi nous définissons dans la suite les **porteurs de types** et **porteurs de termes**, deux signatures linéaires abstraites simples, qui vont nous permettre de transformer une mACG en une composition d'ACG simples.

Décomposition de mACG en ACG simples

Définition 39 (Porteurs)**.** *Soit $\Pi = \langle A, C, \tau^+ \rangle$ une signature surchargée.*

- *Le **porteur de types** $\Sigma_\Pi^{\mathscr{T}} = \langle A, C', \tau_0 \rangle$ de Π est la signature simple où $C' = \biguplus_{c \in C} \{c^\alpha \mid \alpha \in \tau^+(c)\}$ et $\tau_0 : c^\alpha \mapsto \alpha$.*
- *Le **porteur de termes** $\Sigma_\Pi^\Lambda = \langle {}^A/_{\sim_\Pi}, C, \tau_1 \rangle$ de Π est la signature simple où $\tau_1 : c \mapsto [\alpha]_{\sim_\Pi}$ tel que $\alpha \in \tau^+(c)$ est n'importe quel type de c et $[\alpha]_{\sim_\Pi}$ sa classe d'équivalence dans $\sim_\Pi$.*
- *Le réétiquettage de porteur $\mathscr{R}_\Pi : \Sigma_\Pi^{\mathscr{T}} \to \Sigma_\Pi^\Lambda$ de Π est le réétiquettage tel que $\forall c^\alpha \in C', \mathscr{R}_\Pi(c^\alpha) = c$ et $\forall a \in A, \mathscr{R}_\Pi(a) = [a]_{\sim_\Pi}$*

Le domaine $\Sigma_\Pi^{\mathscr{T}}$ de $\mathscr{R}_\Pi$ est appelé porteur de types car il construit les mêmes types implicatifs linéaires que Π mais se débarasse du multitypage en créant une constante annotée par type $\alpha \in \tau^+$.

Le codomaine Σ_Π^Λ de $\mathscr{R}_\Pi$ est appelé le porteur de termes de Π car il construit les mêmes λ-termes linéaires que Π. Cependant, le porteur de termes types les constantes de Π avec des classes d'équivalences de la relation $\sim_\Pi$ plutôt qu'avec des types linéaires construits sur C_Π.

Définition 40 (Signatures $\mathscr{R}$-surchargées)**.** *Soient deux signatures simples $\Sigma_0 = \langle A_0, C_0, \tau_0 \rangle$ et $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$, et soit $\mathscr{R} : \Sigma_0 \to \Sigma_1$ un réétiquettage.*

*On peut définir la signature surchargée de $\mathscr{R}$ (ou signature **$\mathscr{R}$-surchargée**) comme $\Pi_{\mathscr{R}} = \langle A_0, C_1, \tau^+ \rangle$ la signature surchargée de $\tau^+ : c \mapsto \tau_0(\mathscr{R}^{-1}(c))$.*

Définition 41 (Lexiques de types et de termes)**.** *Soit $\mathscr{L}^+ : \Pi \to \Sigma$ un lexique multitypé.*
*Le **lexique de types** $\mathscr{L}^{\mathscr{T}} : \Sigma_\Pi^{\mathscr{T}} \to \Sigma$ de $\mathscr{L}^+$ est défini comme :*

- *$\mathscr{L}^{\mathscr{T}}(\alpha) = \mathscr{L}^+(\alpha)$ pour $\alpha \in \mathscr{T}_{\Sigma_\Pi^{\mathscr{T}}}$,*
- *$\mathscr{L}^{\mathscr{T}}(t) = \mathscr{L}^+ \circ \mathscr{R}_\Pi(t)$ pour $t \in \Lambda(\Sigma_\Pi^{\mathscr{T}})$.*

*Le **lexique de termes** $\mathscr{L}^\Lambda : \Sigma_\Pi^\Lambda \to \Sigma$ est défini comme :*

- *$\mathscr{L}^\Lambda([\alpha]_{\sim_\Pi}) = \mathscr{L}^+(\alpha)$ pour $[\alpha]_{\sim_\Pi} \in \mathscr{T}_{\Sigma_\Pi^\Lambda}$;*
- *$\mathscr{L}^\Lambda(M) = \mathscr{L}^+(M)$ pour $M \in \Lambda(\Sigma_\Pi^\Lambda)$.*

On peut représenter une mACG sous la forme d'un triplet $(\Sigma_\Pi^{\mathscr{T}}, \Sigma_\Pi^\Lambda, \mathscr{R}_\Pi)$ représenté sous forme de diagramme en figure 4.1.

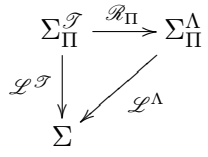

FIGURE 4.1 – Diagramme d'un lexique $\mathscr{L}^+ : \Pi \to \Sigma$ avec Π le triplet $(\Sigma_\Pi^{\mathscr{T}}, \Sigma_\Pi^\Lambda, \mathscr{R}_\Pi)$

Composition en tant que pullback

En encodant deux ACG multitypées $\mathscr{G}_0^+$ et $\mathscr{G}_1^+$ comme dans la figure 4.1, il devient apparent que pour pouvoir les composer, le langage objet des deux ACG simples qui

composent $\mathscr{G}_0^+$ doit être le porteur de types de $\mathscr{G}_1^+$, ce dernier produisant les mêmes λ-termes abstraits que ceux de $\Lambda(\Pi_1)$. La construction que l'on obtient est donc celle de la figure 4.2.

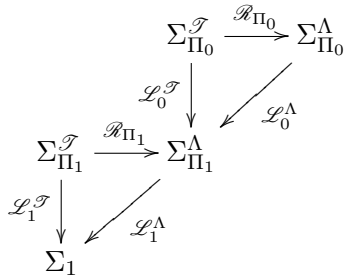

FIGURE 4.2 – Composition de deux ACG multitypées décomposées en ACG simples

Composer nos ACG revient donc à trouver, comme montré dans la figure 4.3, une signature Σ un lexique $\mathscr{L}_* : \Sigma \to \Sigma^{\mathscr{T}}_{\Pi_1}$ et un réétiquetage $\mathscr{R}_* : \Sigma \to \Sigma^{\mathscr{T}}_{\Pi_0}$.

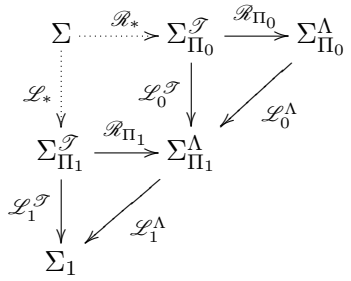

FIGURE 4.3 – Solution au problème de composition

Construction de Kanazawa

Cette signature prend la forme du produit fibré entre $\Sigma^{\mathscr{T}}_{\Pi_0}$ et $\Sigma^{\mathscr{T}}_{\Pi_1}$. Ce produit fibré peut-être formé à l'aide de la construction de Kanazawa, décrite pour la première fois dans Kanazawa (2006). Cette dernière a été créée pour montrer la clôture par intersection avec les encodages d'arbres et de chaînes, mais elle se généralise à toute signature qui est à la fois le langage objet d'un lexique et d'un réétiquettage.

Définition 42 (Produit fibré, Kanazawa, 2006)**.** *Soient Σ, Σ_0 et Σ_1 des signatures simples. Soient $\mathscr{L} : \Sigma_0 \to \Sigma$ un lexique simple et $\mathscr{R} : \Sigma_1 \to \Sigma$ un réétiquettage. Leur produit fibré est la signature $\Sigma_0 \times_{\mathscr{L},\mathscr{R}} \Sigma_1 = \langle A, C, \tau \rangle$ avec le lexique $\mathscr{L}^{\mathscr{R}} : \Sigma_0 \times_{\mathscr{L},\mathscr{R}} \Sigma_1 \to \Sigma_1$ et le réétiquettage $\mathscr{R}^{\mathscr{L}} : \Sigma_0 \times_{\mathscr{L},\mathscr{R}} \Sigma_1 \to \Sigma_0$ tels que :*

— *L'ensemble des types atomiques du produit fibré $\Sigma_0 \times_{\mathscr{L},\mathscr{R}} \Sigma_1$ est le produit fibré des types atomiques de Σ_0 et des types linéaires implicatifs construits sur Σ_1 par $\mathscr{L}$ et $\mathscr{R}$:*

$$A = \{a^\beta \mid a \in A_0, \beta \in \mathscr{T}_{\Sigma_1}, \mathscr{L}(a) = \mathscr{R}(\beta)\}$$

— *L'ensemble des constantes du produit fibré est le produit fibré des constantes de* Σ_0 *et des* λ*-termes linéaires construits sur* Σ_1 *par* $\mathscr{L}$ *et* $\mathscr{R}$ *:*

$$C = \{c_{N:\beta} \mid c \in C_0, \vdash_{\Sigma_1} N : \beta, \mathscr{L}(\vdash_{\Sigma_0} c : \tau_0(c)) = \mathscr{R}(\vdash_{\Sigma_1} N : \beta)\}$$

— $\tau : c_{N:\beta} \mapsto anti(\tau_0(c), \beta)$ *avec* $anti(\alpha_0 \multimap \alpha_1) = anti(\alpha_0, \beta_0) \multimap anti(\alpha_1, \beta_1)$ *et* $anti(a, \beta) = a^\beta$ *pour tout* $a \in A_0$.
— $\mathscr{L}^{\mathscr{R}}$ *oublie la base :* $\forall a^\beta \in A_0, \mathscr{L}^{\mathscr{R}}(a^\beta) = \beta$ *et* $\forall c_{N:\beta} \in C_0, \mathscr{L}^{\mathscr{R}}(c_{N:\beta}) = N$
— $\mathscr{R}^{\mathscr{L}}$ *oublie l'annotation :* $\forall a^\beta \in A_0, \mathscr{R}^{\mathscr{L}}(a^\beta) = a$ *et* $\forall c_{N:\beta} \in C_0, \mathscr{R}^{\mathscr{L}}(c_{N:\beta}) = c$

Ce produit fibré nous permet de définir la composition de mACG comme :

Définition 43 (Composition de mACG)**.** *Soient* $\mathscr{G}_0^+ = \langle \Pi_0, \Sigma_\Pi^\Lambda, \mathscr{L}_0^+, S_0 \rangle$ $\mathscr{G}_1^+ = \langle \Pi_1, \Sigma, \mathscr{L}_1^+, S_1 \rangle$ *deux mACG. Leur composition est l'ACG multitypée* $\mathscr{G}_0^+ \circ \mathscr{G}_1^+ = \langle \Pi_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}, \Sigma, \mathscr{L}_1^+ \circ \mathscr{L}_0^+, \mathscr{W}_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}, S_0 \rangle$ *où* $\Pi_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}$ *est la signature* $(\mathscr{R}_{\Pi_0} \circ \mathscr{R}_{\Pi_1}^{\mathscr{L}^{\mathscr{T}}})$*-surchargée dont le porteur de types est* $\Sigma_{\Pi_0}^{\mathscr{T}} \times_{\Sigma_{\Pi_1}^\Lambda} \Sigma_{\Pi_1}^{\mathscr{T}}$ *et le porteur de termes est* $\Sigma_{\Pi_0}^\Lambda$.

Dans le chapitre suivant, nous décrivons les ACG pondérées, leur multitypage et leur composition.

CHAPITRE 5

ACG pondérées (wACG)

5.1 Semi-anneaux et S-ACG

Définition 44 (Semi-anneau commutatif)**.** *Un semi-anneau est un tuple* $\langle S, +, 0, \cdot, 1\rangle$ *où* $\langle S, +, 0\rangle$ *est un monoïde commutatif (dit additif) et où* $\langle S, \cdot, 1\rangle$ *est un monoïde (dit multiplicatif). La multiplication est distributive sur l'addition et le* 0 *est absorbant pour la multiplication. Le semi-anneau est dit multiplicatif quand le monoïde multiplicatif* $\cdot$ *est commutatif.*

Pour la suite, il est requis du semi-anneau que nous allons manipuler qu'il soit complet. Autrement dit, que l'addition doit s'étendre à une somme infinie. Quand le contexte le permet, nous dénoterons le semi-anneau par l'ensemble S.

Et de manière plus générale, comme nous manipulons des grammaires probabilistes et leur encodage, nous utiliserons l'intervalle réel $[0; 1]$ muni de la multiplication et de l'addition usuelle sur les réels ainsi que de $0_{\mathbb{R}}$ et $1_{\mathbb{R}}$ la plupart du temps

Définition 45 (S-ACG)**.** *Une ACG pondérée sur le semi-anneau S (S-ACG) est un tuple* $\langle \Sigma_{abs}, \Sigma_{obj}, \mathscr{L}, \mathscr{W}, s\rangle$ *où :*

- $\langle \Sigma_{abs}, \Sigma_{obj}, \mathscr{L}, s\rangle$ *est une ACG ;*
- $\mathscr{W} : \Lambda(\Sigma_{abs}) \to S$ *est un morphisme de poids, i.e. un morphisme défini par ses actions sur les constantes abstraites :*

$$\forall u, v \in \Lambda(\Sigma_{abs}), \forall x \in X, \mathscr{W}(uv) = \mathscr{W}(u) \cdot \mathscr{W}(v), \mathscr{W}(\lambda x.u) = \mathscr{W}(u), \mathscr{W}(x) = 1_S.$$

Étant donné un terme abstrait u, $\mathscr{W}(u)$ *est appelé poids de* u*. Le poids d'un terme objet* $t \in \Lambda(\Sigma_{obj})$ *est la somme des poids de ses antécédents par* $\mathscr{L}$ *restreints à ceux dont le type distingué est* s *:*

$$\mathscr{W}(t) = \sum_{\vdash_{\Sigma_{abs}} u : s \in \mathscr{L}^{-1}(t)} \mathscr{W}(u).$$

La définition s'étend aux mACG via le porteur de type, on défini alors une S-mACG comme un tuple $\mathscr{G}^+_S = \langle \Pi, \Sigma, \mathscr{L}^+, \mathscr{W}, S\rangle$ *où* $\langle \Pi, \Sigma, \mathscr{L}^+, S\rangle$ *est une mACG et*

$\mathscr{W} : \Lambda(\Sigma_\Pi^{\mathscr{T}}) \to S$ est un morphisme de poids. Le poids d'un terme abstrait $u \in \Lambda(\Pi)$ est son poids en tant que terme objet du lexique de type :

$$\mathscr{W}(u) = \sum_{v \in \mathscr{R}_\Pi^{-1}(u)} \mathscr{W}(v) \quad \mathscr{W}(t) = \sum_{u:s|s \in S \wedge \mathscr{L}^+(u)=t} \mathscr{W}(u)$$

La définition du poids d'un terme objet peut générer une somme infinie. C'est pourquoi notre semi-anneau doit être complet.

Définition 46 (Composition de wACG)**.** *Soient $\mathscr{G}_0 = \langle \Sigma_0, \Sigma_1, \mathscr{L}_0, \mathscr{W}_0, s_0 \rangle$ et $\mathscr{G}_1 = \langle \Sigma_1, \Sigma_2, \mathscr{L}_1, \mathscr{W}_1, s_1 \rangle$ deux S-ACG.*

Leur composition est la wACG $\mathscr{G}_1 \circ \mathscr{G}_0 = \langle \Sigma_0, \Sigma_2, \mathscr{L}_1 \circ \mathscr{L}_0, \mathscr{W}_{\mathscr{G}_1 \circ \mathscr{G}_0}, s_0 \rangle$ où $\forall u \in \Lambda(\Sigma_0), \mathscr{W}_{\mathscr{G}_1 \circ \mathscr{G}_0}(u) = \mathscr{W}_0(u) \cdot \mathscr{W}_1(\mathscr{L}_0(u))$.

La composition des ACG pondérées s'étend aux ACG pondérées multitypées.

Définition 47 (Composition de S-mACG)**.** *Soient $\mathscr{G}_0^+ = \langle \Pi_0, \Sigma_\Pi^\Lambda, \mathscr{L}_0^+, \mathscr{W}_0, S_0 \rangle$ et $\mathscr{G}_1^+ = \langle \Pi_1, \Sigma, \mathscr{L}_1^+, \mathscr{W}_1, S_1 \rangle$ deux S-mACG.*

Leur composition est la mACG pondérée $\mathscr{G}_0^+ \circ \mathscr{G}_1^+ = \langle \Pi_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}, \Sigma, \mathscr{L}_1^+ \circ \mathscr{L}_0^+, \mathscr{W}_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}, S_0 \rangle$ où $\Pi_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}$ est la signature $(\mathscr{R}_{\Pi_0} \circ \mathscr{R}_{\Pi_1}^{\mathscr{L}^{\mathscr{T}}})$-surchargée, et $\forall u \in \Sigma_{\Pi_0}^{\mathscr{T}} \times_{\Sigma_{\Pi_1}^\Lambda} \Sigma_{\Pi_1}^{\mathscr{T}}, \mathscr{W}_{\mathscr{G}_1^+ \circ \mathscr{G}_0^+}(u) = \mathscr{W}_0(\mathscr{R}_{\Pi_1}^{\mathscr{L}^{\mathscr{T}}}(u)) \cdot \mathscr{W}_1(\mathscr{L}_0^{\mathscr{T}^{\mathscr{R}_{\Pi_1}}}(u))$.

Nous ne nous servons pas de cette composition par la suite, mais il nous semblait important de présenter son existence.

Dans les chapitres suivants, nous nous servirons des notions présentées jusqu'ici pour encoder des grammaires probabilistes dans des ACG et wACG.

CHAPITRE 6

Encodage des automates dans les ACG

Pour chacun des encodages de formalismes de grammaires dans une grammaire catégorielle abstraite, il est généralement suffisant de décomposer ce formalisme entre ses structures abstraites et ses structures objets. Dans la plupart des cas, le langage objet des grammaires sont les chaînes de caractères, mais les structures abstraites sont diverses. C'est donc cette approche que nous employons dans les chapitres suivants pour encoder les formalismes que nous traitons.

6.1 Automates

Les automates ont pour structure sous-jacente les transitions entre les états. Chaque production ou analyse d'un automate s'accompagne ainsi d'une marche dans le graphe dont les états sont les nœuds et les transitions les arêtes.

La signature abstraite de l'encodage d'un automate va donc créer une constante par transition ainsi que des constantes pour ses états d'entrées et ses états acceptants afin de marquer le début et la fin d'une marche.

Définition 48 (Signature abstraite de l'encodage d'un automate)**.** *La signature* Σ_{auto} *de l'encodage d'un automate est le triplet* $\langle A_{auto}, C_{auto}, \tau_{auto}\rangle$ *où :*

- $A_{auto} = \{q_X \mid \forall q_X \in S\} \cup \{q_{init}\}$;
- $C_{auto} = \{c^{q_X}_{q_Y} \mid c \in V, q_X, q_Y \in S, \delta(c, q_X) = q_Y\} \cup \{\#_{q_X} \mid q_X \in F\} \cup \{c^{q_X}_{q_i init} | q_X \in E\}$;
- $\tau_{auto} = \{c^{q_X}_{q_Y} \mapsto q_Y \multimap q_X \mid c^{q_X}_{q_Y} \in C_{auto}\}$.

Nous noterons que le type des constantes représentant une transition de q_X vers q_Y n'est pas $q_X \multimap q_Y$, mais $q_Y \multimap q_X$. Ceci sert simplement à former des λ-termes qui se réalisent en des phrases se lisant de gauche à droite, de l'état initial à l'état final de la marche.

Comme dit plus tôt, dans la plupart des cas, nous générons ou analysons des chaînes de caractères. La signature que nous décrivons dans la définition 49 encode les chaînes de caractères d'un automate.

Définition 49 (Signature objet de l'encodage d'un automate)**.** *La signature Σ_{str} de l'encodage d'un automate est le triplet $\langle A_{str}, C_{str}, \tau_{str}\rangle$ où :*
- $A_{str} = \{o\}$;
- $C_{str} = V \cup \{\#\}$;
- $\tau_{str} = \{c \mapsto o \multimap o \mid \forall c \in V\} \cup \{\# \mapsto o\}$.

La constante # est la constante qui représente la fin d'une chaîne de caractère et le type $o \multimap o$ est le type des chaînes. Par la suite, nous utilisons $\sigma = o \multimap o$ pour nommer ce type.

Définition 50 (Lexique de l'encodage d'un automate dans une ACG)**.** *Le lexique $\mathscr{L}$ de l'encodage d'un automate dans une ACG est le couple $\langle F, G\rangle$ tel que :*
- $F = \{q_X \mapsto o \mid \forall q_X \in A_{auto}\}$;
- $G = \{c^{q_X}_{q_Y} \mapsto c \mid \forall c^{q_X}_{q_Y} \in C_{auto}\} \cup \{\#_{q_X} \mapsto \# \mid \forall \#_{q_X} \in C_{auto}\}$.

Exemple d'automate

Nous pouvons, à l'aide de toutes ces notions, définir une ACG qui encode l'automate de la figure 6.1 qui reconnaît le langage $(a \mid b)^*$ où l'opérateur unaire * est l'étoile de Kleene. Il s'agit du même automate que celui présenté en exemple dans le chapitre 2.

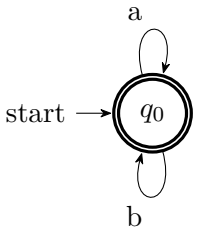

FIGURE 6.1 – Automate reconnaissant la grammaire $(a \mid b)^*$

Pour cela, on commence par définir la signature linéaire abstraite des chaines $\Sigma_{str} = \langle A_{\Sigma_{str}}, C_{\Sigma_{str}}, \tau_{\Sigma_{str}}\rangle$ où :
- $A_{\Sigma_{str}} = \{o\}$ pour pouvoir définir le type des chaines de caractères $\sigma = o \multimap o$;
- $C_{\Sigma_{str}} = \{a, b, \#\}$;
- $\forall s \in C_{\Sigma_{str}}, \tau_{\Sigma_{str}}(s) = \sigma$.

Σ_{str} sera le vocabulaire objet de notre ACG.

Pour définir son vocabulaire abstrait, nous utiliserons la méthode suivante :
- Son unique type atomique est l'état de l'automate q_0 qui sera interprété dans Σ_{str} comme o par le lexique ;
- une constante $\#_{q_0}$ qui dénote le fait que q_0 est un état acceptant. On l'interprétera par $\# : o \in \Sigma_{str}$;
- pour chaque transition $\delta(q_i, a) = q_j$ de notre automate on a une constante δ : $q_j \multimap q_i$ interprétée par $a : \sigma$;

— le type distingué q_0 correspond à notre état initial.
Nous pouvons donc définir une ACG $\mathscr{G}_{ab} = \langle \Sigma_{ab}, \Sigma_{str}, \mathscr{L}_{ab2str}, q_0 \rangle$ où :
— $\Sigma_{ab} = \{\#_{q_0} : q_0, a : q_0 \multimap q_0, b : q_0 \multimap q_0\}$;
— $\Sigma_{str} = \{a : \sigma, b : \sigma, \# : o\}$;
— $\mathscr{L}_{ab2str} = \langle \{q_o \mapsto o\}, \{\#_{q_0} \mapsto \#, a \mapsto a, b \mapsto b\} \rangle$;

6.2 Modèles de Markov Cachés

Les encodages de grammaires dans les ACG ne sont pas uniques. Pour chaque formalisme, il existe plusieurs façons de réaliser son encodage et chacune a des avantages et des inconvénients. Dans le cas des modèles de Markov cachés, un encodage existe déjà et est décrit dans (de Groote, Ludmann, & Pogodalla, s. d.), un article non encore publié.

Encodage existant

Nous présentons ici l'encodage décrit dans cet article.

Définition 51 (HMM dans une ACG de Groote et al., s. d.)**.** *Soit un modèle de Markov caché $\mu = \langle Q, V, A, B, \pi \rangle$. Une première manière d'encoder ce modèle dans une ACG est $\mathscr{G}_{\mu 1} = \langle \Sigma_1, \Sigma_{str}, \mathscr{L}_1, q_{init}, \mathscr{W}_1 \rangle$ où :*
— *La signature abstraite $\Sigma_1 = \langle Q \uplus \{q_{init}\}, C_{\mu 1}, \tau_{\mu 1} \rangle$ où $C_{\mu 1} = \{o^{q_X}_{q_Y} \mid o \in V, q_X \in Q \uplus \{q_{init}\}, q_Y \in Q\} \bigcup \{\#_q \mid q \in Q \uplus \{q_{init}\}\}$ et $\forall o \in V, \forall q_X, q_Y \in Q, \tau_{\mu 1}(o^{q_X}_{q_Y}) = q_Y \multimap q_X, \tau_{\mu 1}(\#_q) = q$;*
— *La signature objet $\Sigma_{str} = \langle \{q\}, V, o \mapsto q \multimap q \rangle$ pour tout $o \in V$;*
— *Le lexique $\mathscr{L}_1$ est $\forall o \in V, \forall q_X, q_Y \in Q, \mathscr{L}_1(o^{q_X}_{q_Y}) = o, \mathscr{L}_1(q_X) = q, \mathscr{L}_1(\#_{q_X}) = \#$;*
— *Le morphisme de poids $\mathscr{W}_1$ est défini comme $\forall o \in V, \forall q_X, q_Y \in Q, \mathscr{W}_1(o^{q_{init}}_{q_Y}) = \pi_{q_Y} \cdot B_{q_Y,o}, \mathscr{W}_1(o^{q_X}_{q_Y}) = A_{q_X,q_Y} \cdot B_{q_Y,o}, \mathscr{W}_1(\#_{q_X}) = 1$;*

Une seconde manière d'encoder les HMM

Nous présentons dans cette partie une seconde manière d'encoder les HMM dans les ACG qui présentent quelques différences vis-à-vis de l'encodage de la partie 6.2. Celle-ci encode des HMM à émissions sur les arcs, strictement équivalents en terme d'expressivité aux HMM à émissions sur les états. Un HMM à émission sur les arcs transforme les états cachés en transitions de l'automate et double le nombre d'états par rapport à l'encodage précédent. La signature abstraite de cet encodage reflète ces changements.

Définition 52 (Signature abstraite de l'encodage d'un HMM à émissions sur les arcs)**.** *La signature abstraite Σ_{HMM_A} d'un HMM à émission sur les arcs est un triplet $\langle A_{HMM_A}, C_{HMM_A}, \tau_{HMM_A} \rangle$ où :*
— *$A_{HMM_A} = Q \uplus \{q_{init}\} \uplus \{q_{X_{Emit}} \mid q_X \in Q\}$;*
— *$C_{HMM_A} = \{\pi_{q_X} \mid q_X \in Q, \pi_{q_X} \neq 0\} \cup \{A^{q_X}_{q_Y} \mid q_X, q_Y \in Q, a_{q_X,q_Y} \neq 0\} \cup \{\#_{q_X} \mid q_X \in Q \uplus \{q_{init}\}\} \uplus \{o_{q_X} \mid o \in V, q_X \in Q, b_{q_X,o} \neq 0\}$;*
— *$\forall q_X, q_Y \in Q, \forall o \in V, \tau_{HMM_A}(\#_{q_X}) = q_{X_{emit}}, \tau_{HMM_A}(\pi_{q_X}) = q_X \multimap q_{init}$, $\tau_{HMM_A}(A^{q_X}_{q_Y}) = q_Y \multimap q_{X_{Emit}}, \tau_{HMM_A}(o_{q_X}) = q_{X_{Emit}} \multimap q_X$.*

La signature Σ_{str} est la même que celle de la définition 51.

Le lexique quand à lui prend en compte les constantes π_X et A_Y^X et les envoie sur la chaîne vide ϵ représentée par $\lambda x.x$.

Définition 53 (Lexique de l'encodage d'un HMM à émissions sur les arcs)**.** *Le lexique* $\mathscr{L}_2$ *est* $\forall q_X, q_Y \in Q, \forall o \in V, \mathscr{L}_2(q_X) = q, \mathscr{L}_2(\#_{q_X}) = \#, \mathscr{L}_2(\pi_{q_X}) = \mathscr{L}_2(A_{q_Y}^{q_X}) = \lambda x.x, \mathscr{L}_2(o_{q_X}) = o$;

Enfin, le morphisme de poids $\mathscr{W}_{HMM_A}$ associe les poids comme suit.

Définition 54 (Morphisme de poids de l'encodage d'un HMM à émissions d'arcs)**.** *Le morphisme de poids* $\mathscr{W}_2$ *est défini par* $\forall q_X, q_Y \in Q, \forall o \in V, \mathscr{W}_2(\#_{q_X}) = 1, \mathscr{W}_2(\pi_{q_X}) = \pi_{q_X}, \mathscr{W}_2(A_{q_Y}^{q_X}) = a_{q_X q_Y}, \mathscr{W}_2(o_{q_X}) = b_{q_X o}$

Définition 55 (Encodage d'un HMM à émissions d'arcs dans une wACG)**.** *Soit un modèle de Markov caché* $\mu = \langle Q, V, A, B, \pi \rangle$*. Un second encodage de* μ *est* $\mathscr{G}_{\mu_2} = \langle \Sigma_{HMM_2}, \Sigma_{str}, \mathscr{L}_2, q_{init} \rangle$ *où :*

- Σ_{HMM_2} *est construit suivant la définition 52 ;*
- Σ_{str} *est construit comme celle de la définition 51 ;*
- $\mathscr{L} : \Sigma_{HMM_2} \to \Sigma_{str}$ *est construit selon la définition 53 ;*
- $\mathscr{W} : \Sigma_{HMM_2} \to [0; 1]$ *est construit selon la définition 54.*

Il existe en λ-calcul une notation canonique pour les chaînes de caractères

Définition 56 (Concaténation de chaînes $(+)$)**.** *La concaténation de deux chaînes* c_1 *et* c_2 *est la composition de fonction :*

$$\lambda f g. \lambda z. f(g\ z).$$

On l'écrit $c_1 + c_2$*.*

En découle que la chaîne vide ϵ *est l'identité* $\lambda x.x$

Pour la suite de ce mémoire, nous utiliserons cette définition pour décrire la concaténation de symboles.

Dans la suite, nous montrons que notre encodage des HMM à émission d'arcs est un encodage fidèle au HMM qu'il encode.

Proposition 1. *Soit* $\mu = \langle Q, V, A, B, \pi \rangle$ *un HMM et* $\mathscr{G}_\mu$ *son encodage.*

On a $\forall q_1 \in Q, \forall (O, X) \in V^n \times Q^n$ *:*

$$\mathscr{L}_\mu(\psi_\mu^{x_0}(O, X)\#_{q_n}) = o_1 + \cdots + o_n$$
$$\mathscr{W}_\mu((\pi_{q_0}\ (\psi_\mu^{x_0}(O, X)\ \#_{q_n}))) = P(O, X \mid q_1, \mu)$$
$$\mathscr{W}_\mu((o_1 + \cdots + o_n)\#) = P(O \mid X)$$

Où

$$\psi_\mu^{q_1}((O, X)) = \lambda x.(o_{1_{q_1}}(A_{q_2}^{q_1}(o_{2_{q_2}}(\ldots(o_{n-1_{q_{n-1}}}(A_{q_n}^{q_{n-1}}(o_{n_{q_n}} x)))\ldots)))).$$

Et tout terme abstrait de $\mathcal{A}(\mathscr{G}_{HMM_2})$ *est de la forme :*

$$\pi_{q_1}(\lambda x.(o_{1_{q_1}}(A_{q_2}^{q_1}(o_{2_{q_2}}(\ldots(o_{n-1_{q_{n-1}}}(A_{q_n}^{q_{n-1}}(o_{n_{q_n}} x)))\ldots))))\#_{q_n})$$

Démonstration. On commence par montrer :

$$\forall (O, X), O = o_1 + o_2 + \cdots + o_n, X = q_1 q_2 \dots q_n,$$

$$\exists! t = \lambda x.t', \mathscr{L}_{HMM_2}(t) = o_1 + o_2 + \cdots + o_n \wedge \mathscr{W}(t) = P(O, X \mid q_1, \mu)$$

Par induction sur n, le couple $(O, X) = (\epsilon, \epsilon)$ force $t' = x$, car x doit apparaître dans t', t étant linéaire. La seule constante qui est antécédant de la chaîne vide # par $\mathscr{L}_{HMM_2}$ est $\#_{q_{init}}$ et on a bien $((\lambda x.x)\#_{q_{init}}) \in \mathcal{A}(\mathscr{G})$ et $\mathscr{W}(\#_{q_{init}}) = 1$.

Dans le cas inductif, où $O = O' \cdot o_{n+1}$ et $X = X' \cdot q_{n+1}$. Nous avons deux cas possibles pour le type de $\lambda x.t'$:

— Soit $\vdash_{\Sigma_{HMM_2}} \lambda x.t' : q_{n_{Emit}} \multimap q_1$ et il existe par définition de notre encodage un unique λ-terme $(A^{q_n}_{q_{n+1}} o_{n+1_{q_{n+1}}} x)$ tel que

$$\begin{aligned}
\lambda x.\mathscr{L}_{HMM_2}(t'(A^{q_n}_{q_{n+1}} o_{n+1_{q_{n+1}}} x)) &= \lambda x.\mathscr{L}_{HMM_2}(t')(\mathscr{L}_{HMM_2}(A^{q_n}_{q_{n+1}} o_{n+1_{q_{n+1}}})x) \\
&= \lambda x.\mathscr{L}_{HMM_2}(t')(\mathscr{L}_{HMM_2}((A^{q_n}_{q_{n+1}})\mathscr{L}_{HMM_2}(o_{n+1_{q_{n+1}}}))x) \\
&= \lambda x.\mathscr{L}_{HMM_2}(t')(((\lambda y.y)\mathscr{L}_{HMM_2}(o_{n+1_{q_{n+1}}}))x) \\
&= \lambda x.\mathscr{L}_{HMM_2}(t')(\mathscr{L}_{HMM_2}(o_{n+1_{q_{n+1}}})x) \\
&= o_1 + o_2 + \cdots + o_n + o_{n+1}
\end{aligned}$$

On a bien

$$\begin{aligned}
\mathscr{W}((\lambda x.t')((A^{q_n}_{q_{n+1}} o_{n+1_{q_{n+1}}} x))) &= \mathscr{W}(t') \times \mathscr{W}(A^{q_n}_{q_{n+1}}) \times \mathscr{W}(o_{n+1_{q_{n+1}}}) \\
&= P(O', X' \mid q_1, \mu) \times a_{q_n q_n+1} \times b_{q_{n+1} o_{n+1}} \\
&= P(O, X \mid q_1, \mu).
\end{aligned}$$

— Soit $\vdash_{\Sigma_{HMM_2}} \lambda x.t' : q_{n+1} \multimap q_1$ et le terme le plus profond de t' ne peut être que $A^{q_n}_{q_{n+1}}$, car c'est par définition de l'encodage le seul terme de type $q_{n+1} \multimap T$ (avec T un type quelconque, q_n dans ce contexte) qui peut s'appliquer à un $o_{q_{n+1}}$, ou x avec $x : q_{n+1}$.

Il existe donc, toujours par définition de l'encodage, une unique constante $o_{n+1_{q_{n+1}}} \in C_{HMM_2}$ telle que

$$\begin{aligned}
\lambda x.\mathscr{L}_{HMM_2}(t'(o_{n+1_{q_{n+1}}} x)) &= \lambda x.\mathscr{L}_{HMM_2}(t')(\mathscr{L}_{HMM_2}(o_{n+1_{q_{n+1}}})x) \\
&= o_1 + o_2 + \cdots + o_n + o_{n+1}
\end{aligned}$$

On a bien

$$\begin{aligned}
\mathscr{W}((\lambda x.t')(o_{n+1_{q_{n+1}}} x)) &= \mathscr{W}(t') \times \mathscr{W}(A^{q_n}_{q_{n+1}}) \times \mathscr{W}(o_{n+1_{q_{n+1}}}) \\
&= P(O', X' \mid q_1, \mu) \times a_{q_n q_n+1} \times b_{q_{n+1} o_{n+1}} \\
&= P(O, X \mid q_1, \mu).
\end{aligned}$$

Pour toute séquence $(O, X) \in V^n \times Q^n$ où q_0 est un état initial, il existe (par définition de $\mathscr{G}_{HMM_2}$) une unique constante $\pi_{q_0} : q_0 \multimap q_{init}$ et comme tous les états sont acceptants dans le modèle que nous avons donné, il existe un $\#_{q_n} : q_n$ quel que soit q_n.

Ainsi, pour tout λ-terme t issu de (O, X), si O est reconnu par μ alors

$$\pi_{q_0} \ (t \ \#_{q_n}) \in \mathcal{A}(\mathscr{G}_{HMM_2}).$$

Et enfin $\forall O \in V^*$ on a bien :

$$\begin{aligned}
\mathscr{W}((o_1 + o_2 + \cdots + o_n)\#) &= \mathscr{W}(\mathscr{L}_{HMM_2}(\pi_{q_1} \ (t \ (\#_{q_n})))) \\
&= \sum_{\forall X \in Q^n} \mathscr{W}(\pi_{q_1}) \times \mathscr{W}(t) \\
&= \sum_{\forall X \in Q^n} \pi_{q_1} \times b_{q_1,o_1} \times P(O, X \mid q_1, \mu) \\
&= \sum_{\forall X \in Q^n} P(O, X \mid q_{init}, \mu) \\
&= P(O \mid \mu)
\end{aligned}$$

□

Nous avons montré la correction de notre wACG ainsi que sa forte équivalence avec le HMM qu'elle encode.

Il peut être intéressant de comparer les deux encodages décrits ci-dessus entre eux. Le second, celui crée pour ce mémoire est plus fidèle au HMM encodé dans le sens où il y a une correspondance un pour un entre les constantes abstraites de l'ACG et les champs des matrices du HMM.

Cependant, ce dernier requiert plus de calcul, la plupart des constantes abstraites se réalisant en l'identité, ce qui cause une augmentation massive du nombre de β-réductions nécessaires au calcul d'un terme du langage objet de l'ACG.

Il est également possible de mobiliser les ACG multitypées pour encoder les HMM en supprimant les annotations d'état des constantes et en assignant plusieurs types à la constante A, la constante π et à chaque constante o représentant un symbole terminal o. Les poids sont ensuite assignés comme décrit dans la définition 45.

CHAPITRE 7

Encodage des CFG dans les ACG

7.1 Grammaires non contextuelles

L'encodage d'une grammaire non contextuelle tel que défini dans de Groote (2001), de Groote et Pogodalla (2004) s'appuie sur le fait que les arbres de dérivations sont les structures abstraites manipulées par les CFG et encode ainsi les règles de cette grammaire comme des constantes abstraite. Voici comment cet encodage est défini dans de Groote et Pogodalla (2004).

Dans les définitions suivantes, on traite d'une grammaire non contextuelle quelconque $G = \langle N, T, P, S\rangle$

Définition 57 (Signature abstraite de l'encodage d'une CFG dans une ACG)**.** *La signature abstraite Σ_{rules} de l'encodage d'une CFG est un triplet $\langle A_{rules}, C_{rules}, \tau_{rules}\rangle$ où :*

- $A_{rules} = \{X \mid X \in N\}$;
- $C_{rules} = \{\rho_r \mid r \in P\}$;
- $\tau_{rules} : C_{rules} \to \mathscr{T}_{A_{rules}}$ *et pour tout $\rho_r \in C_{rules}$ avec r une règle de la forme $a \to \alpha_1 a_1 \alpha_2 a_2 \dots \alpha_n a_n$ telle que $a, \alpha_1, \alpha_2, \dots, \alpha_n \in N$ et $a_1, a_2, \dots, a_n \in T^*$, on a $\tau_{rules}(\rho_r) = \alpha_1 \multimap \alpha_2 \multimap \dots \multimap \alpha_n \multimap a$.*

Définition 58 (Signature objet de l'encodage d'une CFG dans une ACG)**.** *La signature objet Σ_{str} de l'encodage d'une CFG dans une ACG est le triplet $\langle A_{str}, C_{str}, \tau_{str}\rangle$ où :*

- $A_{str} = \{o\}$;
- $C_{str} = \{t \mid t \in T\}$ *l'ensemble des constantes issues des symboles terminaux de la grammaire ;*
- $\tau_{str} : C_{str} \to \mathscr{T}_{A_{str}}$ *tel que pour tout $c \in C_{str}, \tau(c) = \sigma$ où σ est le type string $o \multimap o$.*

Définition 59 (λ-terme $[\![r]\!]$ issu d'une règle de production)**.** *Si r est une règle de production de la forme $a \to a_0 \alpha_1 a_1 \dots \alpha_n a_n$ avec $a, \alpha_1, \alpha_2, \dots, \alpha_n \in N$ et $a_0, a_1, a_2, \dots, a_n \in T^*$ alors $[\![r]\!]$ est un λ-terme linéaire défini par :*

$$\lambda x_1 \dots x_n . a_0 + x_1 + a_1 + x_2 + a_2 + \dots + x_n + a_n.$$

Définition 60 (Lexique de l'encodage d'une CFG dans une ACG)**.** *Le lexique $\mathscr{L}_{rules}$ de l'encodage d'une CFG dans une ACG est un couple $\langle F, G\rangle$ où :*

- $F = \{X \mapsto \sigma \mid X \in A_{rules}\}$ *est un réétiquetage des types atomiques de A_{rules} vers le type $\sigma = o \multimap o$ de $\mathscr{T}_{A_{str}}$;*
- $G = \{\rho_r \mapsto [\![r]\!] \mid r \in P\}$

Définition 61 (Encodage d'une CFG dans une ACG)**.** *Une CFG G encodée dans une ACG $\mathscr{G}_{CFG}$ est un quadruplet $\langle \Sigma_{rules}, \Sigma_{str}, \mathscr{L}, S\rangle$ où :*

- Σ_{rules} *est construite comme dans la définition 57 ;*
- Σ_{str} *est construite comme dans la définition 58 ;*
- $\mathscr{L}_{rules}$ *est construit comme dans la définition 60 ;*

Exemple d'encodage

La figure 7.1 est une CFG décrivant une grammaire non contextuelle ambigüe permettant à la préposition "with" d'être rattachée soit à un groupe verbal, soit à un groupe nominal.

S → NP VP	P → with
PP → P NP	V → saw
VP → V NP	Det → a
VP → VP PP	N → man
NP → NP PP	N → telescope
NP → Det N	NP → John

FIGURE 7.1 – CFG

Cette dernière générerait donc la grammaire catégorielle abstraite $\mathscr{G}_{ppr} = \langle \Sigma_{ppr}, \Sigma_{str}, \mathscr{L}_{ppr}, S\rangle$ dont la signature abstraite est donnée en figure 7.2.

$$
\begin{aligned}
\Sigma_{ppr} = \quad & \{\ \rho_S \mapsto NP \multimap VP \multimap S,\\
& \rho_{PP} \mapsto P \multimap NP \multimap PP,\\
& \rho_{VP} \mapsto V \multimap NP \multimap VP,\\
& \rho_{VP_{PP}} \mapsto VP \multimap PP \multimap VP,\\
& \rho_{NP} \mapsto Det \multimap N \multimap NP,\\
& \rho_{NP_{PP}} \mapsto NP \multimap PP \multimap NP,\\
& \rho_{With} \mapsto P, \rho_{Saw} \mapsto V,\\
& \rho_A \mapsto Det, \rho_{Man} \mapsto N,\\
& \rho_{Telescope} \mapsto N, \rho_{John} \mapsto NP\ \}.
\end{aligned}
$$

FIGURE 7.2 – Signature abstraite Σ_{rules}

La signature Σ_{str} de cette ACG est celle de la figure 7.3 et assigne simplement le type σ à tous les symboles terminaux de la grammaire.

Et enfin, le lexique est défini dans la figure 7.4.

$$\Sigma_{str} = \{with \mapsto \sigma, saw \mapsto \sigma, a \mapsto \sigma, man \mapsto \sigma, telescope \mapsto \sigma, John \mapsto \sigma\}$$

FIGURE 7.3

$$\begin{aligned}\mathscr{L}_{ppr} = \quad & \{ \{ X \mapsto o \mid X \in \{S, N, NP, V, VP, P, PP\}\}, \\ & \{ \rho_S \mapsto \lambda x.\lambda y.x + y, \\ & \rho_P \mapsto \lambda x.\lambda y.x + y, \\ & \rho_{VP} \mapsto \lambda x.\lambda y.x + y, \\ & \rho_{VP_{PP}} \mapsto \lambda x.\lambda y.x + y, \\ & \rho_{NP} \mapsto \lambda x.\lambda y.x + y, \\ & \rho_{NP_{PP}} \mapsto \lambda x.\lambda y.x + y, \\ & \rho_{With} \mapsto with, \rho_{Saw} \mapsto saw, \\ & \rho_A \mapsto a, \rho_{Man} \mapsto man, \\ & \rho_{Telescope} \mapsto telescope, \rho_{John} \mapsto John\}\}\end{aligned}$$

FIGURE 7.4 – Lexique $\mathscr{L}_{ppr}$

7.2 Grammaires non contextuelles probabilistes

Comme décrit en partie 2.2, une pCFG est munie d'une fonction de poids qui associe à chaque règle une probabilité. L'encodage des CFG de la partie précédente encode chaque règle sous forme de constantes abstraites. Il suffit donc pour définir ce morphisme de poids d'envoyer chaque constante abstraite de l'ACG à la probabilité assignée à la règle qui lui correspond dans la pCFG.

Définition 62 (Morphisme de poids d'une ACG encodant une pCFG)**.** *Le morphisme de poids $\mathscr{W}$ d'une wACG $\mathscr{G} = \langle \Sigma_{rules}, \Sigma_{str}, \mathscr{L}_{rules}, \mathscr{W}, S\rangle$ encodant une pCFG $G = \langle N, T, P, S, \mu\rangle$ est défini comme*

$$\mathscr{W} = \{\rho_r \mapsto x \mid \rho_r \in C_{rules}, x = \mu(r)\}.$$

Nous montrons maintenant que la définition 62 du morphisme de poids est correcte dans le sens où elle se comporte comme la pCFG qu'elle encode.

Proposition 2. *Pour toute dérivation d d'une pCFG G quelconque produisant un mot w de poids p, il existe un λ-terme t de $\mathcal{A}(\mathscr{G})$ (où $\mathscr{G}$ est l'encodage de G dans une wACG) qui se réalise en w et dont le poids par le morphisme $\mathscr{W}$ est p.*

Pour nous simplifier la démonstration de cette proposition, nous commencerons par poser le lemme suivant.

Lemme 1. *L'encodage d'une grammaire non contextuelle G dans une ACG $\mathscr{G}$ par la méthode décrite en partie 7.1 implique que G est en équivalence forte avec $\mathscr{G}$.*

La preuve de ce lemme se trouve dans de Groote et Pogodalla (2004).

Démonstration. Ce lemme nous dit que pour chaque arbre de dérivation d de G produisant un mot w il existe exactement un λ-terme $t \in \mathcal{A}(\mathscr{G})$ dont la structure est la même que celle de cet arbre et qui se réalise en w. i.e, pour chaque règle r de d, il y a exactement une constante dans t modélisant r.

Or la structure de $\mathscr{G}$ ne change pas par le simple ajout du morphisme de poids $\mathscr{W}$.

Il est ainsi suffisant de montrer que $\mathscr{W}$ associe bien à chaque terme le produit du poids de chaque constante de ce terme.

Par définition du morphisme de poids $\mathscr{W}$ d'une ACG, le poids d'un terme est entièrement défini par le produit du poids de ses constantes. □

Exemple d'encodage

La figure 7.5 étend la grammaire de la figure 7.1 de poids.

S → NP VP	1.0	P → with	1.0
PP → P NP	1.0	V → saw	1.0
VP → V NP	0.7	Det → a	1.0
VP → VP PP	0.3	N → man	0.5
NP → NP PP	0.4	N → telescope	0.5
NP → Det N	0.5	NP → John	0.1

FIGURE 7.5 – CFG

L'encodage restant le même, nous ne décrivons en figure 7.6 que le morphisme de poids qui va directement associer chaque terme abstrait à la probabilité qui lui est associée dans la grammaire de la figure 7.5.

$$
\begin{aligned}
\mathscr{W} = \quad & \{\ \rho_S \mapsto 1.0, \\
& \rho_P \mapsto 1.0, \\
& \rho_{VP} \mapsto 0.7, \\
& \rho_{VP_{PP}} \mapsto 0.3, \\
& \rho_{NP} \mapsto 0.5, \\
& \rho_{NP_{PP}} \mapsto 0.4, \\
& \rho_{With} \mapsto 1.0, \rho_{Saw} \mapsto 1.0, \\
& \rho_A \mapsto 1.0, \rho_{Man} \mapsto 0.5, \\
& \rho_{Telescope} \mapsto 0.5, \rho_{John} \mapsto 0.1\}.
\end{aligned}
$$

FIGURE 7.6 – Morphisme de poids

CHAPITRE 8

Encodage des TAG dans les ACG

L'encodage des grammaires d'arbres adjoints demande une architecture différente des grammaires que nous avons vues jusqu'ici. En effet, si jusqu'ici nous pouvions nous contenter d'un ACG qui encode l'entièreté de notre grammaire, ici il nous faut modéliser les arbres dérivés et les arbres de dérivations. Pour ça, nous présentons l'architecture détaillée dans de Groote (2002). À noter qu'une architecture plus complète existe dans Pogodalla (2017) détaillant également comment extraire la sémantique d'une TAG sous forme de formules logiques. Cette architecture-ci passant par l'ajout d'un réétiquettage, nous ne nous attarderons pas dessus, mais nous soulignons qu'il est simple de passer de l'encodage de 2002 à celui de 2017.

8.1 Grammaires d'arbres adjoints

Contrairement à Pogodalla (2017), nous commençons par décrire l'encodage des arbres de dérivations dans une ACG $\mathscr{G}_{TAG} = \langle \Sigma_{TAG}, \Sigma_{trees}, \mathscr{L}_{TAG}, S_S \rangle$.

Soit $G = \langle N, V, I, A, S \rangle$ une grammaire TAG.
Nous définissons $\mathscr{G}_{TAG} = \langle \Sigma_{TAG}, \Sigma_{trees}, \mathscr{L}_{TAG}, S_S \rangle$ comme suit :

Définition 63 (Signature abstraite Σ_{TAG} de l'encodage d'une TAG)**.** *La signature Σ_{TAG} est un triplet $\langle A_{TAG}, C_{TAG}, \tau_{TAG} \rangle$ où :*

- *L'ensemble des types atomiques A_{TAG} est composé de deux copies de l'ensemble N des symboles non-terminaux. Pour tout $\alpha \in N$ on ajoute un α_S et un α_A dans l'ensemble A_{TAG} ;*
- *Pour chaque arbre T de I dont la racine est étiquetée de α, les nœuds de substitution sont étiquetés par les $\beta_1, \ldots, \beta_n$ et dont les nœuds internes sont étiqutées par les $\gamma_1, \ldots, \gamma_m$, on crée une constante :*

$$c_T : \gamma_{1A} \multimap \cdots \multimap \gamma_{mA} \multimap \beta_{1S} \multimap \cdots \multimap \beta_{nS} \multimap \alpha_S;$$

- *Pour chaque arbre T' de A dont la racine est étiquetée de α, les nœuds de substitution sont étiquetés par les $\beta_1, \ldots, \beta_n$ et dont les nœuds internes sont étiqutées*

par les $\gamma_1, \ldots, \gamma_m$, on crée une constante :

$$c_{T'} : \gamma_{1A} \multimap \cdots \multimap \gamma_{mA} \multimap \beta_{1S} \multimap \cdots \multimap \beta_{nS} \multimap \alpha_A \multimap \alpha_A;$$

— *Pour chaque symbole non-terminal $\alpha \in N$, on crée une constante I_α de type α_A*

La signature objet Σ_{trees} nous permet de représenter les arbres de la grammaire G pour la représenter nous avons besoin d'un alphabet gradué :

Définition 64 (Alphabet gradué Pogodalla, 2017)**.** *Un alphabet gradué est une paire $\mathscr{F}_a = \langle \mathscr{F}, arity \rangle$ où :*

— *$\mathscr{F}$ est un ensemble de symboles de fonction ;*

— *$arity : \mathscr{F} \to \mathbb{N}$ est une fonction qui à chaque symbole de $\mathscr{F}$ associe le nombre d'arguments qu'elle prend en entrée.*

On peut ensuite définir la signature comme suit :

Définition 65 (Signature abstraite Σ_{trees} Pogodalla, 2017)**.** *La signature Σ_{trees} est un triplet $\langle A_T, \mathscr{F}, \tau_{trees} \rangle$ où :*

— *$A_T = \{T\}$ contient l'unique type T, type des arbres ;*

— *$\tau_{trees}^{\mathscr{F}_a}$ est une fonction qui à chaque constante X telle que $arity(X) = n$ associe le type $\underbrace{T \multimap \cdots \multimap T}_{n \text{ fois}} \multimap T$. Si $arity(X) = 0$ alors $\tau_{trees}^{\mathscr{F}_a}(X) = T$.*

Enfin, le lexique $\mathscr{L}_{TAG}$ de notre grammaire doit former des arbres dérivés dans Σ_{trees} à partir des arbres de dérivations de Σ_{TAG}.

Les arbres initiaux se retrouvent représentées comme des arbres dans Σ_{trees} là où les arbres auxiliaires se retrouvent représentés comme des fonctions sur les arbres. Autrement dit pour chaque non-terminal $\alpha \in N$ on a $\mathscr{L}_{TAG}(\alpha_S) = T$ et $\mathscr{L}_{TAG}(\alpha_A) = T \multimap T$. Les constantes I_α introduites plus tôt servent à compléter un λ-terme de Σ_{TAG} en signifiant l'absence d'adjonction. On a donc $\mathscr{L}_{TAG}(I_\alpha) = T \multimap T$ pour pouvoir les passer en argument d'un nœud d'adjonction.

Exemple sur la grammaire TAG de la partie 2

Si on reprend l'exemple de grammaire TAG de la figure 2.2, on a donc la grammaire suivante :

$$\begin{aligned}\Sigma_{TAG} = \langle &\{S_A, S_S, NP_S, N_A, VP_A, V_A, Adj_A\},\\ &\{c_{people}, c_{eat}, c_{peanuts}, c_{roasted}, I_S, I_N, I_{VP}, I_V, I_{Adj}\},\\ &\{c_{eat} \mapsto S_A \multimap VP_A \multimap V_A \multimap NP_S \multimap NP_S \multimap S_S,\\ &c_{people} \mapsto N_A \multimap NP_S,\\ &c_{peanuts} \mapsto N_A \multimap NP_S,\\ &c_{roasted} \mapsto N_A \multimap Adj_A \multimap N_A \multimap N_A, \} \cup \{I_X \mapsto X_A \multimap X_A \mid X \in \{S, N, VP, V, Adj\}\rangle.\end{aligned}$$

La signature Σ_{trees} de cet exemple est :

$$\begin{aligned}\Sigma_{trees} = \langle &\{T\}, \{S_2, NP_1, VP_2, V_1, N_1, N_2, Adj_1, people, eat, roasted, peanuts\}\\ &\{X \mapsto T \mid X \in \{people, eat, roasted, peanuts\}\} \cup\\ &\{X \mapsto T \multimap T \mid X \in \{NP_1, V_1, N_1, Adj_1\}\} \cup\\ &\{X \mapsto T \multimap T \multimap T \mid X \in \{NP_2, VP_2, N_w\}\}\rangle.\end{aligned}$$

Et enfin son lexique $\mathscr{L}_{TAG}$ est :

$$\begin{aligned}\mathscr{L}_{TAG} = \langle &\{X \mapsto T \mid X \in \{S_S, NP_S\}\} \cup\\ &\{X \mapsto T \multimap T \mid X \in \{S_A, N_A, VP_A, V_A, Adj_A\}\},\\ &\{c_{people} \mapsto \lambda F.(NP_1(F(N_1 people))),\\ &c_{peanuts} \mapsto \lambda F.(NP_1(F(N_1 peanuts))),\\ &c_{eat} \mapsto, \lambda F.\lambda G.\lambda H.\lambda x.\lambda y.F(S_2\ x(G\ (VP_2\ (H\ (V_1\ eat))y)))\\ &c_{roasted} \mapsto \lambda F.\lambda G.\lambda H \lambda x.F(N_2\ (G\ (Adj_1\ roasted)))(H\ x), \} \cup\\ &\{I_X \mapsto \lambda x.x \mid X \in \{S, N, VP, V, Adj\}\}\rangle\end{aligned}$$

Des arbres vers les chaînes

Il nous faut maintenant transformer les arbres en chaînes de caractères. Pour ceci, nous définissons une seconde ACG $\mathscr{G}_{yield} = \langle \Sigma_{trees}, \Sigma_{str}, \mathscr{L}_{yield}, T\rangle$. Le vocabulaire abstrait étant déjà défini il nous faut définir le vocabulaire objet de cette ACG ainsi que son lexique.

Le vocabulaire objet Σ_{str} est défini exactement de la même manière que pour les CFG en utilisant la définition 58.

Enfin, le lexique $\mathscr{L}_{yield}$ de $\mathscr{G}_{yield}$ est défini de la manière suivante :

Définition 66 (Lexique de l'ACG $\mathscr{G}_{yield}$)**.** *Le lexique $\mathscr{L}_{yield}$ de l'ACG $\mathscr{G}_{yield}$ est un couple $\langle F, G\rangle$ où :*

— *$F = \{T \mapsto o \multimap o\}$, un arbre est interprété comme une chaîne ;*

— $G = \{a \mapsto a \mid a \in T\} \cup \{\alpha_i \mapsto \lambda x_1 \ldots \lambda x_i . x_1 + \cdots + x_i\}$. *G envoie les constantes de* Σ_{trees} *représentant des terminaux sur celles de* Σ_{str} *représentant les mêmes terminaux et les non-terminaux d'arité i sur un combinateur d'arité i.*

Il nous suffit donc pour finir de définir l'encodage d'une grammaire TAG dans une ACG de dire qu'il s'agit de la composition d'une ACG $\mathscr{G}_{TAG}$ et d'une ACG $\mathscr{G}_{yield}$:

$$\mathscr{G}_{yield} \circ \mathscr{G}_{TAG}$$

Suite de l'exemple d'encodage

Pour continuer notre exemple, la signature Σ_{str} correspondante est

$$\begin{aligned}\Sigma_{str} = \langle \{o\}, \{&people, eat, roasted, peanuts\}, \\ &\{a \mapsto o \multimap o \mid a \in \{people, eat, roasted, peanuts\}\}\rangle\end{aligned}$$

Et le lexique $\mathscr{L}_{yield} : \Sigma_{trees} \to \Sigma_{str}$ est

$$\begin{aligned}\mathscr{L}_{yield} = \langle &\{T \mapsto o \multimap o\}, \\ &\{people \mapsto people, peanuts \mapsto peanuts, eat \mapsto eat, roasted \mapsto roasted, \} \cup \\ &\{X \mapsto \lambda x.x \mid X \in \{NP_1, V_1, N_1, Adj_1\}\} \cup \\ &\{X \mapsto \lambda x.\lambda y.x + y \mid X \in \{S_1, VP_2, N_2\}\}\rangle\end{aligned}$$

8.2 Grammaires d'arbres adjoints probabilistes

L'encodage des grammaires d'arbres adjoints probabilistes ne diffère que peu de l'encodage des TAG classiques. L'architecture décrite dans la partie 8.1 reste la même, sauf pour l'ACG $\mathscr{G}_{TAG}$ dont la signature Σ_{TAG} et le lexique $\mathscr{L}_{TAG}$ sont modifiés comme expliqué ci-dessous.

L'encodage repose sur l'ajout à la signature Σ_{TAG} de constantes représentant les évènements d'adjonction et de substitution afin de leur assigner un poids. Un poids est également assigné à tous les arbres, afin de représenter la probabilité qu'ont ces arbres d'être la racine d'un arbre de dérivation.

Représentation des évènements de substitution et d'adjonction

Nous cherchons dans un premier temps à rendre impossible la substitution et l'adjonction d'un arbre sur un autre sans passer par une constante représentant un évènement, comme mentionné plus haut.

La manière la plus simple de faire ceci est, pour chaque type atomique de chaque type implicatif de chaque arbre, d'annoter ce type de son arbre de provenance ainsi que de numéroter sa place dans le type implicatif en question.

Plus formellement la signature abstraite Σ_{pTAG} de l'encodage d'une pTAG dans une wACG est définie comme suit

Définition 67 (Signature abstraite Σ_{pTAG} de l'encodage d'une pTAG)**.** *La signature Σ_{pTAG} est un triplet $\langle A_{pTAG}, C_{pTAG}, \tau_{pTAG}\rangle$ où :*

- *Pour chaque arbre T de I dont la racine est étiquetée de α, les nœuds de substitution sont étiquetés par les $\beta_1, \ldots, \beta_n$ et dont les nœuds internes sont étiquetées par les $\gamma_1, \ldots, \gamma_m$, on crée une constante :*

$$c_T : \gamma_{A1}^{(c_T)} \multimap \cdots \multimap \gamma_{Am}^{(c_T)} \multimap \beta_{S1}^{(c_T)} \multimap \cdots \multimap \beta_{Sn}^{(c_T)} \multimap \alpha_{Sn+1}^{(c_T)};$$

- *Pour chaque arbre T' de A dont la racine est étiquetée de α, les nœuds de substitution sont étiquetés par les $\beta_1, \ldots, \beta_n$ et dont les nœuds internes sont étiquetées par les $\gamma_1, \ldots, \gamma_m$, on crée une constante :*

$$c_{T'} : \gamma_{A1}^{(c_{T'})} \multimap \cdots \multimap \gamma_{Am}^{(c_{T'})} \multimap \beta_{S1}^{(c_{T'})} \multimap \cdots \multimap \beta_{Sn}^{(c_{T'})} \multimap \alpha_{Am+1}^{(c_{T'})} \multimap \alpha_{Am+2}^{(c_{T'})};$$

- *Pour toute paire d'arbre (α, α') de la grammaire pTAG où la constante c_α est de type $\gamma_1 \multimap \cdots \multimap \gamma_{Si} \ldots \gamma_{Sj} \multimap \gamma_n$ et $c_{\alpha'}$ est de type $\beta_1 \multimap \cdots \multimap \beta_m$, nous créons pour chaque tout $i \leq k \leq j$ une constante $S_{m\ k}^{(\alpha')\ (\alpha)} : \beta_m^{(\alpha')} \multimap \gamma_k^{(\alpha)}$ si $\gamma = \beta$ pour représenter la substitution de l'arbre α' sur l'arbre α sur le site d'adjonction représenté par le nœud encodé par $\gamma_{Sk}^{(\alpha)}$.*
- *Pour toute paire d'arbre (α, α') de la grammaire pTAG où la constante c_α est de type $\gamma_{A1} \ldots \gamma_{Ai} \multimap \gamma_{i+1} \multimap \ldots \gamma_n$ et α' est de type $\beta_1 \multimap \cdots \multimap \beta_{Am}$, nous créons pour chaque tout $1 \leq k \leq i$ une constante $A_{m\ k}^{(\alpha')\ (\alpha)} : \beta_{Am}^{(\alpha')} \multimap \gamma_{Ak}^{(\alpha)}$ si $\gamma = \beta$ pour représenter l'adjonction de l'arbre α' sur l'arbre α sur le site d'adjonction représenté par le nœud encodé par $\gamma_{Ak}^{(\alpha)}$.*
- *Nous créons finalement une constante I de type β_A ainsi qu'une constante $A_k^{(none)\ (\alpha)} : \beta_A \multimap \beta_{Am}^{(\alpha)}$ pour tout type β_{Am} modélisant un nœud $\star$ d'un arbre auxiliaire et pour tout $\beta_{Am}^{(\alpha)}$ modélisant un site d'adjonction.*

La constante c_{eat} de la signature Σ_{TAG} de l'exemple précédent devient donc :

$$c_{eat} : S_{A1}^{(c_{eat})} \multimap VP_{A2}^{(c_{eat})} \multimap V_{A3}^{(c_{eat})} \multimap NP_{S1}^{(c_{eat})} \multimap NP_{S2}^{(c_{eat})} \multimap S_{S3}^{(c_{eat})}$$

Et la constante $c_{roasted}$ de cette même signature devient :

$$c_{roasted} : N_{A1}^{(c_{roasted})} \multimap Adj_{A2}^{(c_{roasted})} \multimap N_{A3}^{(c_{roasted})} \multimap N_{A4}^{(c_{roasted})}$$

Enfin, nous définissons le lexique $\mathscr{L}_{pTAG}$ comme suit :

Définition 68 (Lexique $\mathscr{L}_{pTAG}$ de l'encodage d'une pTAG dans une wACG)**.** *Le lexique $\mathscr{L}_{pTAG} : \Sigma_{pTAG} \to \Sigma_{derived\ trees}$ de l'encodage d'une pTAG dans une wACG est un couple $\langle F, G\rangle$ tel que :*

- *F envoie tous les types $\beta_{Ak}^X \in A_{pTAG}$ sur le type $T \multimap T$ de $\mathscr{T}^{A_{derived\ trees}}$ tous les types $\beta_{Sk}^X \in A_{pTAG}$ sur le type T de $A_{derived\ trees}$;*

— *G a un comportement similaire à celui du lexique de $\mathscr{L}_{TAG}$ à ceci près qu'en plus d'envoyer les I_Y^X sur $\lambda x.x$, les $A_Z^{(X)(Y)}$ et les $S_Z^{(X)(Y)}$ sont également envoyés sur $\lambda x.x$.*

Le morphisme de poids $\mathscr{W}_{pTAG}$ est ensuité defini de la manière suivante :

Définition 69 (Morphisme de poids $\mathscr{W}_{pTAG}$ de l'encodage d'une pTAG dans une wACG)**.** *Le morphisme de poids $\mathscr{W}_{pTAG} : C_{pTAG} \to [0;1]$ envoie les constantes de C_{pTAG} dans l'intervalle des probabilités de la manière suivante :*

— *Pour tout arbre $c_\alpha \in C_{pTAG}$ tel que $\alpha \in I$, $\mathscr{W}(c_\alpha) = P_I(\alpha)$;*

— *Pour toute constante $S_x^{(\alpha)(\alpha')}$ représentant l'évènement de substitution de α' dans l'arbre α au nœud d'adresse η représenté par le type $\beta_x^{(\alpha)}$, $\mathscr{W}(S_x^{(\alpha)(\alpha')}) = S(\alpha, \alpha', \eta)$;*

— *Pour toute constante $A_x^{(\alpha)(\alpha')}$ représentant l'évènement d'adjonction de α' dans l'arbre α au nœud d'adresse η représenté par le type $\beta_x^{(\alpha)}$, $\mathscr{W}(A_x^{(\alpha)(\alpha')}) = A(\alpha, \alpha', \eta)$;*

— *Pour tout autre constante $c \in c_{pTAG}$, $\mathscr{W}(c) = 1$;*

Le travail sur l'encodage des pTAG dans les mACG est en cours et la preuve d'équivalence entre les pTAG et leur encodage est encore à l'état de conjecture. Il s'agit d'une des pistes pour l'encodage des pTAG et d'autres restent encore à explorer. Notamment, en utilisant le multitypage pour encoder les adjonctions plutôt que de passer par des constantes de coercion de type comme décrit ci-dessus.

CHAPITRE 9

Conclusion

Dans ce mémoire, nous avons étudié l'encodage de grammaires stochastiques dans les grammaires catégorielles abstraites. En particulier, nous avons défini l'encodage des modèles de Markov cachés, des grammaires non contextuelles probabiliste et des grammaires d'arbres adjoints probabilistes dans les wACG.

Nous avons prouvé la forte équivalence dans le cas des HMM et des pCFG, mais demeure la question de celle de l'encodage des pTAG.

Ce nouveau champ des ACG pondérées et multitypées ouvre de nombreuses questions qui n'attendent qu'à être explorée et il nous semble pertinent de dire que ces encodages ne sont qu'un premier pas dans un domaine vaste.

Comment obtenir les poids des grammaires en les entraînant directement dans l'ACG ? Existe-t-il des encodages plus efficaces à l'analyse ou à la génération ? Le parsing dans les ACG pondérées est-il algorithmiquement moins complexe en temps et en espace que les grammaires encodées ?

Une piste inexplorée qui a émergée durant ce stage porte également sur la possibilité d'encoder des probabilités dépendantes d'adjonction et de substitution à l'aide du multitypage. La pertinence de cette possibilité n'a pas non plus été explorée.

L'implementation dans l'outil ACGtk (outil permettant de développer des ACG pour analyser et générer du texte expérimentalement) des notions sous-jacentes à ces encodages pourrait grandement aider à tester ces notions sur de grandes grammaires.

Références

Abeillé, A. (1993). *Les Nouvelles Syntaxes : Grammaires d'unification et analyse du français*. France : Armand Colin.

Barendregt, H. (1985). *The lambda calculus : Its syntax and semantics* (2nd edition éd.). Studies in Logic and the Foundations of Mathematics 103.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, *29*(1), 47–58. Consulté le 2024-06-08, sur `http://www.jstor.org/stable/410452`

Chiang, D. (2000, octobre). Statistical parsing with an automatically-extracted Tree Adjoining Grammar. In *Proceedings of the 38th annual meeting of the association for computational linguistics* (pp. 456–463). Hong Kong : Association for Computational Linguistics. Consulté sur `https://aclanthology.org/P00-1058` doi : 10.3115/1075218.1075276

de Groote, P. (2001, juillet). Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter* (p. 148-155). Toulouse, France, France. Consulté sur `https://inria.hal.science/inria-00100529` (Colloque avec actes et comité de lecture. internationale.) doi : 10.3115/1073012.1073045

de Groote, P. (2002, mai). Tree-Adjoining Grammars as Abstract Categorial Grammars. In R. Frank (Ed.), *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)* (p. 145-150). Venezia, Italy. Consulté sur `https://inria.hal.science/hal-04120630`

de Groote, P., Ludmann, P., & Pogodalla, S. (s. d.). *Multityped and Weighted Abstract Categorial Grammars.* personal communication.

de Groote, P., & Pogodalla, S. (2004). On the expressive power of Abstract Categorial Grammars : Representing context-free formalisms. *Journal of Logic, Language and Information*, *13*(4), 421-438. Consulté sur `https://inria.hal.science/inria-00112956` (http ://www.springerlink.com/content/1572-9583/) doi : 10.1007/s10849-004-2114-x

Gorn, S. (1967). Explicit definitions and linguistic dominoes. In J. Hart & S. Takasu (Eds.), *Systems and computer science* (pp. 77–115). Toronto : University of Toronto Press. Consulté le 2024-06-09, sur `https://doi.org/10.3138/9781487592769-008` doi : doi:10.3138/9781487592769-008

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2000). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Joshi, A. K., Levy, L. S., & Takahashi, M. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences*, *10*(1), 136-163. Consulté sur `https://www.sciencedirect.com/science/article/pii/S0022000075800195` doi : https://doi.org/10.1016/S0022-0000(75)80019-5

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing : An introduction to natural language processing, computational linguistics, and speech recognition* (1st éd.). USA : Prentice Hall PTR.

Kanazawa, M. (2006). Abstract families of abstract categorial languages. *Electronic Notes in Theoretical Computer Science*, *165*, 65-80. Consulté sur `https://www.sciencedirect.com/science/article/pii/S1571066106005147` (Proceedings

of the 13th Workshop on Logic, Language, Information and Computation (WoLLIC 2006)) doi : https://doi.org/10.1016/j.entcs.2006.05.037

Lambek, J. (1958). The mathematics of sentence structure. *The American Mathematical Monthly*, *65*(3), 154–170. Consulté sur `https://doi.org/10.1080/00029890.1958.11989160` doi : 10.1080/00029890.1958.11989160

Ludmann, P., Pogodalla, S., & de Groote, P. (2022, septembre). Multityped Abstract Categorial Grammars and Their Composition. In *WoLLIC 2022 - 28th International Workshop on Logic, Language, Information, and Computation* (Vol. 13468, p. 105-122). Iaşi, Romania : Springer International Publishing. Consulté sur `https://inria.hal.science/hal-03781596` doi : 10.1007/978-3-031-15298-6_7

Manning, C., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

Moot, R., & Retoré, C. (2012). *The Logic of Categorial Grammars : A Deductive Account of Natural Language Syntax and Semantics* (Vol. 6850). Berlin, Heidelberg : Springer. Consulté le 2024-03-14, sur `http://link.springer.com/10.1007/978-3-642-31555-8` doi : 10.1007/978-3-642-31555-8

Pogodalla, S. (2017). A syntax-semantics interface for Tree-Adjoining Grammars through Abstract Categorial Grammars. *Journal of Language Modelling*, *5*(3), 527-605. Consulté sur `https://inria.hal.science/hal-01242154` doi : 10.15398/jlm.v5i3.193

Resnik, P. (1992). Probabilistic Tree-Adjoining Grammar as a framework for statistical natural language processing. In *COLING 1992 volume 2 : The 14th International Conference on Computational Linguistics*. Consulté sur `https://aclanthology.org/C92-2065`

Yoshinaga, N., Miyao, Y., & Tsujii, J. (2002). A formal proof of strong equivalence for a grammar conversion from ltag to hpsg-style. In *Proceedings of the sixth international workshop on tree adjoining grammar and related frameworks (tag+ 6)* (pp. 187–192).