

# Génération de texte avec les grammaires catégorielles abstraites et la théorie sens-texte

Marie Cousin

▶ **To cite this version:**

Marie Cousin. Génération de texte avec les grammaires catégorielles abstraites et la théorie sens-texte. Informatique [cs]. 2022. hal-03942766

**HAL Id: hal-03942766**
**https://inria.hal.science/hal-03942766**

Submitted on 17 Jan 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Grenoble INP – ENSIMAG
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

# Rapport de Projet de Fin d'Etudes

Effectué au LORIA
Laboratoire Lorrain de Recherche en Informatique et ses Applications

# Génération de texte avec les grammaires catégorielles abstraites et la théorie sens-texte

Cousin Marie
3e année – Filière ISI

01 mars 2022 – 31 août 2022
(25 semaines)

**LORIA**
615 rue du Jardin Botanique
54506 VANDŒUVRE-LES-NANCY

**Responsable de stage**
Sylvain POGODALLA
**Tuteur de l'école**
Olivier MULLER

**Résumé**

La théorie sens-texte est une théorie linguistique permettant de décrire la correspondance entre le sens et le texte d'un énoncé par un outil formel qui simule l'activité langagière d'un locuteur natif. Elle utilise le concept clé de paraphrase lors de cette génération, à l'aide notamment des fonctions lexicales. Les grammaires catégorielles abstraites sont, elles, un formalisme grammatical permettant d'encoder d'autres formalismes grammaticaux, et permettent notamment le passage de représentations sémantiques à des représentations syntaxiques. Ce présent rapport étudie d'une part comment utiliser les grammaires catégorielles abstraites pour la génération de texte selon le modèle de la théorie sens-texte, et explicite d'autre part une implémentation possible pour ce faire, du niveau sémantique à celui de syntaxe de surface de la théorie sens-texte.

**Mots clés**— théorie sens-texte, grammaires catégorielles abstraites

# Remerciements

Je tenais à remercier tout particulièrement tous les membres de l'équipe Sémagramme pour leur accueil formidable, leur enthousiasme et leur convivialité. Mon stage s'est très bien passé, en grande partie grâce à la bonne entente qui règne dans l'équipe.

Je voudrais aussi remercier Sylvain Pogodalla et Philippe de Groote, ainsi que Maxime Amblard et Bruno Guillaume de m'avoir permis d'effectuer mon stage dans l'équipe, et de m'avoir permis de choisir un sujet de stage qui me correspond, presque "sur-mesure".

De plus, je voudrais adresser un énorme merci à Sylvain Pogodalla, mon tuteur, qui a été compréhensif, pédagogue, patient, et qui m'a accompagné tout au long du stage avec le sourire et bienveillance.

Je voudrais aussi remercier Valentin, Vincent, Pierre, Hee-Soo, Priyansh, Gabriel, Gwenaelle, Wenjun et Fanny, avec qui je partageais le bureau, ainsi que Siyana et Lisa, qui sont devenus des amis et grâce à qui mon stage s'est si bien passé.

# Table des matières

**1 Introduction** **1**

**2 Structure d'accueil** **1**
2.1 Le LORIA . . . 1
2.2 L'équipe Sémagramme . . . 2

**3 Problématique** **2**
3.1 Contexte . . . 2
3.1.1 Génération automatique de texte . . . 3
3.1.2 Collocations . . . 3
3.1.3 Fonctions lexicales . . . 4
3.2 Problème posé . . . 4
3.2.1 Objectifs . . . 4
3.2.2 État de l'art . . . 5

**4 Grammaires Catégorielles abstraites et ACGtk** **5**
4.1 Grammaires Catégorielles Abstraites . . . 5
4.1.1 Quelques définitions introductives . . . 6
4.1.2 Les ACGs . . . 6
4.1.3 Applications des ACGs . . . 7
4.2 ACGtk . . . 7

**5 Théorie Sens-Texte** **8**
5.1 Présentation générale . . . 8
5.2 Les 7 niveaux de représentation . . . 9
5.2.1 SemR : la représentation sémantique . . . 9
5.2.2 DSyntR : la représentation syntaxique profonde . . . 10
5.2.3 SSyntR : la représentation syntaxique de surface . . . 10
5.3 Les modules de transition . . . 11
5.3.1 Le module sémantique : SemR $\leftrightarrow$ DSyntR . . . 11
5.3.2 Le module syntaxique profond : DSyntR $\leftrightarrow$ SSyntR . . . 12
5.4 Le Dictionnaire Explicatif Combinatoire . . . 13

**6 Solutions proposées** **13**
6.1 Aperçu de l'implémentation réalisée . . . 14
6.2 Principes généraux mis en oeuvre . . . 15
6.3 Modélisation de la sémantique . . . 17
6.3.1 Objectifs . . . 17
6.3.2 Implémentation . . . 19
6.3.3 Problèmes rencontrés . . . 20
6.4 Modélisation de la syntaxe profonde . . . 23
6.4.1 Objectifs . . . 23
6.4.2 Représentation syntaxique profonde . . . 25
6.4.3 Paraphrase . . . 26
6.4.4 Fonctions lexicales . . . 27
6.5 Modélisation de la syntaxe de surface . . . 28
6.5.1 Objectifs . . . 28
6.5.2 Implémentation . . . 28
6.5.3 Problèmes rencontrés . . . 31

**7 Evaluation** **31**
7.1 Échantillons de test . . . 31
7.2 Méthodologie . . . 32
7.3 Perspectives . . . 32

**8 Analyse des résultats** **32**
8.1 Bilan . . . 32
8.2 Limitations et perspectives . . . 33
8.2.1 Limitations . . . 33
8.2.2 Perspectives . . . 34

**9 Avancement** **34**

**10 Impressions** **35**
10.1 Cadre de travail et journée type . . . 35
10.2 Apports sur le plan professionnel, technique et personnel . . . 36

**11 Impact environnemental et sociétal** **36**
11.1 Impact environnemental personnel de votre PFE . . . 36
11.2 Impact global du projet . . . 37
11.2.1 Impact environnemental du projet . . . 37
11.2.2 Impact sociétal du projet . . . 39
11.3 Politique de la structure d'accueil . . . 39

**12 Conclusion** **40**

**A Compléments sur la TST** **45**
A.1 SemR : la représentation sémantique . . . 45
A.2 DSyntR : la représentation syntaxique profonde . . . 46
A.3 SSyntR : la représentation syntaxique de surface . . . 46
A.3.1 DMorphR : la représentation morphologique profonde . . . 47
A.3.2 SMorphR : la représentation morphologique de surface . . . 47
A.3.3 DPhonR : la représentation phonologique profonde . . . 47
A.3.4 SPhonR : la représentation phonologique de surface . . . 48
A.3.5 Le module syntaxique de surface : SSyntR $\leftrightarrow$ DMorphR . . . 48

**B Compléments sur les FLs** **49**

**C Compléments sur les ACGs** **50**
C.1 Exemple d'ACG simple . . . 50
C.2 Axiomes et règles d'inférences et exemple d'application . . . 50

**D Exemple de paraphrase syntaxique pour la phrase « *Alain is calm* »** **51**

**E Paraphrase syntaxique et arbres de paraphrase générés** **52**

**F Exemple d'un script de test : « *John assasinate Mary.* »** **53**

**G Diagramme de Gantt** **55**

**H Extraits de code issus de ma modélisation** **56**

# 1 Introduction

La théorie sens-texte (TST, Mel'čuk et al. 2012) est une théorie linguistique permettant de décrire la correspondance entre le sens d'un énoncé et sa représentation sous forme textuelle par un outil formel qui simule l'activité langagière d'un locuteur natif. La direction privilégiée par cette théorie est la direction du sens vers le texte. Cet outil formel, le modèle sens-texte (MST), est composée de sept niveaux de représentation ; les niveaux de représentation sémantique, syntaxique profonde, syntaxique de surface, morphologique profonde, morphologique de surface, phonologique profonde et phonologique de surface. Ils représentent chacun les phrases par un ensemble de structures, dont la principale est un graphe ou un arbre de dépendance selon le niveau, et de modules de transition pour passer de l'un à l'autre de ces niveaux. La paraphrase et les fonctions lexicales, décrites en sections 5 et 3.1.3 respectivement sont au centre de ces transitions. Je me suis focalisée lors de ce PFE sur les niveaux de représentation sémantique, syntaxique profonde et syntaxique de surface uniquement.

L'équipe Sémagramme dans laquelle j'effectue mon PFE étudie les langues naturelles et comment les modéliser, et a entre autres développé différents outils pour ce faire, tel qu'ACGtk (cf. section 4.2), qui permet l'utilisation des ACGs. Les grammaires catégorielles abstraites (ACGs, de Groote 2001) sont un formalisme grammatical au sein duquel on peut encoder d'autres formalismes grammaticaux, basé sur le $\lambda$-calcul. Ainsi, si l'on souhaite, comme dans la correspondance sens-texte décrit par la TST, encoder une relation plusieurs à plusieurs, on peut "découper" cette relation en plusieurs étapes, plus simples à encoder (des relations bijectives ou surjectives par exemple), et les composer par la suite. On retrouve alors une simulation de la relation plusieurs à plusieurs initiale, mais qui est en réalité une composition d'autres relations.

Le but de mon PFE est d'étudier la TST et ses différentes structures ainsi que les modules de transition et les règles qui en font partie en utilisant les outils et approches développés par l'équipe Sémagramme, en me focalisant sur la modélisation des collocations et des fonctions lexicales.

# 2 Structure d'accueil

J'ai effectué ce stage au sein de l'équipe Sémagramme au LORIA à Vandœuvre-lès-Nancy. Dans les sections suivantes je présente brièvement le LORIA et Sémagramme.

## 2.1 Le LORIA

Le LORIA (Laboratoire Lorrain de Recherche en Informatique et ses Applications) est l'UMR 7503, soit une Unité Mixte de Recherche. Elle commune à l'INRIA (Institut National en sciences et technologies du numérique), au CNRS (Centre National de la Recherche Scientifique) et à l'Université de Lorraine (UL).

Le LORIA a été créé en 1997 et a pour mission la recherche fondamentale et appliquée en sciences informatiques. Membre de la Fédération Charles Hermite (regroupant trois laboratoires de recherche en mathématiques et science et technologies de l'information et de la communication de Lorraine), le LORIA fait également partie du pôle scientifique AM2I (Automatique, Mathématiques, Informatique et leurs interactions) de l'Université de Lorraine.

Le LORIA est dirigé par Jean-Yves Marion, et plus de 400 personnes y travaillent. Il comprend 29 équipes de recherche, dont 15 en commun avec l'Inria, réparties dans 5 départements scientifiques :

- Algorithmique, calcul, image et géométrie, dont le responsable est Sylvain Lazard,
- Méthodes formelles, dont le responsable est Horatiu Cirstea,
- Réseaux, systèmes et services, dont le responsable est Ye-Qiong Song ;
- Traitement automatique des langues et des connaissances, dont le responsable est Bruno Guillaume,
- Systèmes complexes, intelligence artificielle et robotique, dont le responsable est Jean-Baptiste Mouret.

En outre, différents services sont proposés, comme le restaurant du laboratoire, la médiathèque, les salles de réunion et les salles de détentes, ou la salle de musique, afin que le cadre de travail soit agréable.

## 2.2 L'équipe Sémagramme

L'équipe Sémagramme au sein de laquelle j'effectue mon stage est dirigée par Philippe de Groote, et est composée de membres permanents (7), tels que mon tuteur Sylvain Pogodalla, et de stagiaires (4), doctorants (9), ingénieurs (2) et d'un post-doctorant. C'est une équipe commune à l'Inria et au LORIA. Elle travaille sur la définition et le développement d'outils et de modèles pour l'analyse sémantique d'énoncés et de discours en langue naturelle, basés sur la logique.

Parmi les outils développés par Sémagramme, il y a les outils ACGtk (dont je parle plus précisément en section 4.2) et Grew, qui permet la réécriture de structures linguistiques. Il permet par exemple d'appliquer des règles locales de transformation de graphes, si les structures linguistiques considérées sont des graphes.

# 3 Problématique

Le contexte scientifique et linguistique de mon stage est décrit en section 3.1. J'explicite ensuite en section 3.2 le problème auquel mon stage doit répondre, puis décris les objectifs attendus en section 3.2.1, et fais enfin un rapide état de l'art (section 3.2.2).

## 3.1 Contexte

Ce stage s'inscrit dans un contexte multidisciplinaire, liant informatique et linguistique. En effet, parmi les différents domaines d'application et de recherche en informatique se trouve le traitement automatique des langues naturelles (TALN, ou en anglais NLP (Natural Language Processing)). La reconnaissance de texte ou la traduction automatique sont par exemple des applications de ce domaine. Seulement, qui dit langue naturelle dit aussi ressources linguistiques. Par exemple, l'une des ressource linguistique la plus connue et la plus utilisée est pour moi un dictionnaire papier traditionnel.

Le sujet de ce stage étant la génération de texte avec les grammaires catégorielles abstraites et la théorie sens-texte, je me suis intéressée au sous domaine du TALN qu'est la génération automatique de texte, que je présente rapidement en section 3.1.1, ainsi qu'aux grammaires catégorielles abstraites, et à la théorie sens-texte (cf. section 5). Les grammaires catégorielles abstraites sont un formalisme grammatical permettant de représenter la grammaire de la langue naturelle. Je présente ce formalisme grammatical en section 4.1 avec plus de détails. Or, pour générer du texte, un simple dictionnaire contenant des mots et des définitions ne suffit pas forcément. En effet, la TST utilise un dictionnaire complexe, le dictionnaire explicatif combinatoire (cf. section 5.4). Il a la particularité de non seulement définir les entrée lexicales avec précision, mais également d'expliciter comment elles se combinent avec les autres entrées lexicales, et leurs lien. En effet, la langue naturelle est composée de mots simples, mais également de synonymes, de mots polysémiques (doit-on les considérer comme deux mots différents, ou comme un seul ?), et de phénomènes linguistiques plus complexes, tels que les collocations (cf. partie 3.1.2).

Afin de générer du texte, à l'aide des grammaires catégorielles abstraites et de la TST, j'ai donc eu besoin de représenter les phénomènes linguistiques décrits par la TST à l'aide d'outils informatiques permettant l'utilisation des ACGs. Je décris dans la sous-sections 3.1.2 quelques phénomènes linguistiques en question, et comment ils peuvent être modélisés conceptuellement en section 3.1.3 par des fonctions lexicales.

En outre, les aspects sémantiques et syntaxiques de la langue sont au cœur du traitement des langues, et occupent une place très importante au sein de la TST (Mel'čuk et al. 2012 ; Mel'čuk et al. 2013), et donc également dans ce projet.

**Définition 3.1.1 (Polguère 2003)** *La **sémantique** (d'un énoncé) désigne les sens (de cet énoncé) et leur organisation au sein des messages que l'on peut exprimer.*

**Remarque 3.1.2** *Pour parler du sens véhiculé par une unité lexicale (UL)* A *, j'utilise la notation* $^{\langle}.^{\rangle}$*, empruntée à Polguère et Mel'čuk et al. :* $^{\langle}A^{\rangle}$ *est le sens de l'UL* A *.*

**Définition 3.1.3 (Polguère 2003)** *La **syntaxe** (d'un énoncé) désigne la structure des phrases (de cet énoncé).*

**Exemple 3.1.4** *La phrase* « La pomme mange un camion. » *est sémantiquement absurde, mais syntaxiquement juste. En effet, la construction de la phrase (sujet, verbe, complément d'objet direct) est correcte, mais cette phrase n'a aucun sens.*

### 3.1.1 Génération automatique de texte

Le terme de génération de texte est un sous domaine du TALN, et est utilisé pour décrire un processus lors duquel du texte est produit. Ce texte doit être le plus naturel possible.

**Remarque 3.1.5** *Traditionnellement, lors de ce processus, le logiciel utilisé choisit quelle phrase produire. L'une des particularité de la TST est qu'elle a été pensée pour aider le locuteur non natif d'une langue à choisir comment exprimer le sens qu'il veut communiquer (Mel'čuk et al. 2012 ; Mel'čuk et al. 2013). Cette théorie va donc produire un ensemble de possibilités, et le locuteur choisira parmi cet ensemble. Je ne me suis donc pas attelée au problème du choix de la phrase à générer, car j'ai cherché à générer toutes les phrases possibles exprimant un même sens.*

Le défi majeur de ce projet est donc de générer du texte naturel, de manière automatique, et de la manière la moins coûteuse possible. Trois aspects entrent donc en compte : le modèle utilisé doit produire un texte naturel, l'implémentation utilisée doit permettre une automatisation de la génération, et cette automatisation doit être peu coûteuse en temps et complexité.

Concernant le caractère naturel du texte généré, j'ai travaillé sur la TST (voir section 5), qui permet au texte généré d'être naturel, et qui est basée sur le concept clé de paraphrase. Cependant, indépendamment du modèle linguistique, de nombreux phénomènes lexicaux et expressions de la langue considérée peuvent poser problème, telles que les collocations par exemple (cf. partie 3.1.2). En effet, « *une pluie torrentielle* » est une paraphrase de « *une forte pluie* » ou « *une grosse pluie* », mais « *une chute torrentielle* » pour « *une grosse chute* » n'est pas correct.

### 3.1.2 Collocations

Les collocations sont un type d'expressions, et se retrouvent très fréquemment dans les textes, oraux comme écrits. La plupart des expressions obéissent au principe de compositionnalité sémantique, à savoir que le sens d'une expression est la composition du sens des éléments qui la composent (Polguère 2003) (cf. exemple 3.1.6). Cependant, deux principaux types d'expressions désobéissent à ce principe ; les locutions (ou expressions idiomatiques), et les collocations. Pour ces expressions, le choix des mots employés n'est pas totalement libre (cf. définition 3.1.7).

**Exemple 3.1.6** *Les phrases* « Je mange une pomme. », « Jean aime Marie. » *ou encore* « Je pars jeudi en Bretagne avec Caroline en train. » *obéissent au principe de compositionnalité sémantique.*

**Définition 3.1.7** *[Polguère 2003] L'expression* AB *(ou* BA *) formée des unités lexicales* A *et* B *est une* ***collocation*** *si, lors de l'étape de formation de cette expression,* A *est sélectionnée librement pour son sens* ‘$A$’ *et* B *est sélectionnée pour exprimer un sens* ‘$C$’ *en fonction de* A *.*
*L'UL sélectionnée pour son sens (ici* A *) est appelée* ***base*** *de la collocation alors que la seconde (ici* B *) est appelée* ***collocatif****.*

**Exemple 3.1.8** *Les expressions* « peur bleue » *(* « [peur]$_{\text{A}}$ [bleue]$_{\text{B}}$ » *),* « pleuvoir des cordes » *(* « [pleuvoir]$_{\text{A}}$ [des cordes]$_{\text{B}}$ » *) ou encore* « être enceinte jusqu'aux yeux » *(* « [être enceinte]$_{\text{A}}$ [jusqu'aux yeux]$_{\text{B}}$ » *) sont des collocations.*

Comme on peut le constater, les collocations n'obéissent à aucune règle systématique, et on ne peut pas échanger les collocatifs de deux bases différentes pour un même caractère exprimé. En effet, « *une trompette tonitruante* » et « *un moteur rugissant* » sont deux expressions valides exprimant toutes deux l'intensité d'un bruit produit, mais « *une trompette rugissante* » et « *un moteur tonitruant* » ne le sont pas. Les collocations ont ainsi trois propriétés (Polguère 2003) :

- il y a des collocations dans toutes les langues,
- elles sont omniprésentes dans les textes, oraux comme écrit,
- leur formation semble arbitraire, elles ne peuvent pas se traduire mot à mot d'une langue à l'autre et sont très difficiles à acquérir pour un locuteur non natif.

L'usage des collocations est alors utile, pour générer des expressions proches de ce qu'un locuteur natif aurait utilisé.

### 3.1.3 Fonctions lexicales

Les fonctions lexicales (FLs) sont des outils pour représenter des liens entre unités lexicales, et permettent donc de représenter (entre autres) les collocations. J'en fais ici une brève présentation, vous pourrez trouver de plus amples détails dans l'annexe B et dans (Mel'Čuk et Polguère 2021) ou encore (Mel'čuk et al. 2015).

**Définition 3.1.9 (Mel'Čuk et Polguère 2021)** *Une **fonction lexicale** (FL) $f$ est une fonction qui s'applique à une UL $L$. Une FL indique une modification (précision, complément ou altération par exemple) du sens de $L$. La valeur $f(L)$ de cette fonction est un ensemble de tous les choix alternatifs d'ULs pour la modification apportée à $L$.*

**Exemple 3.1.10** *`magn`(PLUIE ) pourra renvoyer* {forte, torrentielle}*, où `magn` est la FL indiquant l'intensité. De même, `anti`(CALME ) pourra renvoyer* {énervé, agité}*, où `anti` est la FL indiquant le contraire.*

La TST utilise beaucoup les FLs (Mel'Čuk et Polguère 2021) lors de s transitions de la représentation sémantique (la représentation du sens) à la représentation syntaxique de surface.

## 3.2 Problème posé

Les ACGs (de Groote 2001) sont des formalismes grammaticaux basés sur le $\lambda$-calcul grâce auxquels on peut décrire d'autres formalismes grammaticaux. Elles ont l'avantage de définir des structures de dérivation (ou langages abstraits) et offrent la possibilité de spécifier comment les interpréter en des structures de surface (ou langages objets) (voir section 4.1 pour plus de détails).

L'équipe Sémagramme a développé l'outil ACGtk (en OCaml), qui permet d'encoder les ACGs. Les ACGs utilisent les $\lambda$-termes. Elles permettent de généraliser les arbres, chaînes de caractères et d'écrire certains graphes. En effet, les ACGs sont très fortement typées, ce qui est très bien adapté à la manipulation rigoureuse de divers objets, comme les structures linguistiques. En outre, les ACGs permettent de faire correspondre un langage d'arbre avec un langage de chaînes de caractère (Pogodalla 2017a), soit de faire le lien entre deux structures linguistiques différentes.

La TST fait intervenir sept niveaux de représentation, chacun étant représenté par une structure de données spécifique : le premier est représenté principalement par un graphe, les deux suivant par des arbres, et les 4 derniers par des chaînes de caractères. La TST met donc en relation différentes structures. Les ACGs permettant cette mise en relation entre des arbres et des chaînes de caractères, elles paraissent tout à fait appropriées à la description et à la modélisation de la TST.

En outre ACGtk possède la propriété de ne pas modifier la structure de départ pour la transformer en celle de sortie, mais d'en créer une nouvelle à partir de celle d'entrée : les deux structures coexistent donc. Cette propriété se retrouve dans la TST, où les structures de chaque niveau de représentation sont construites à partir de la structure précédente, sans toutefois la modifier. L'utilisation des ACGs et d'ACGtk semble donc particulièrement appropriée.

### 3.2.1 Objectifs

Le but de ce projet est dans un premier temps d'étudier les structures linguistiques proposées par la TST pour les différents niveaux de représentation de la langue (sémantique, syntaxique profonde, syntaxique de surface, morphologique profonde, morphologique de surface, phonologique profonde et phonologique de surface), ainsi que les modules de transition entre ces structures.

Dans un second temps, ce projet a pour but d'utiliser les approches et outils développées dans l'équipe Sémagramme afin de modéliser la TST, au moins jusqu'au niveau de syntaxe de surface ou au niveau morphologique profond. Concernant la validation des solutions théoriques, je pouvais utiliser les logiciels Grew et ACGtk, pour implémenter ces solutions et les tester.

En particulier, les aspects de formalisation et specification des transformations, la prise en compte des ressources lexicales nécessaires et la modélisation des collocations sont trois points importants de mon stage. De plus, les ACGs ont la propriété avantageuse d'être utilisables dans les deux sens de leur interprétation : du langage abstrait au langage objet, et inversement (i.e. du texte au sens et du sens au texte). Exploiter cette propriété est un autre point important du stage.

Aussi, le but final de ce projet est d'obtenir un modèle fonctionnant sur des exemples non triviaux couvrant un certain panel des expressions linguistiques : il doit y avoir parmi les phrases générées des

exemples de collocations, certaines (au moins) doivent utiliser les FLs, et d'autres (si ce n'est pas toutes) permettre la paraphrase au cours de leur génération.

### 3.2.2 État de l'art

Jusqu'à présent, les ACGs n'ont jamais été utilisées pour modéliser la TST. Cependant, de nombreux travaux ont été réalisés et publiés du côté de la TST d'une part, et du côté des ACGs et d'ACGtk.

La littérature autour de la TST et de ses applications ou modélisations est assez riche. Si Igor Mel'čuk a beaucoup travaillé dessus, et publié récemment trois volumes explicatifs complets résumant cette théorie linguistique (Mel'čuk et al. 2012 ; Mel'čuk et al. 2013 ; Mel'čuk et al. 2015), d'autres auteurs se sont aussi attelés à la résumer, tels que Jasmina Milićević qui a écrit un guide explicatif très court (une trentaine de pages) de la TST (Milićević 2006), ou Lidija Iordanskaja qui explicite la paraphrase lors de la génération de texte avec la TST (Iordanskaja, Kittredge et Polguère 1991).

En outre, les travaux de Leo Wanner sur une application de la TST à des fins pratiques (générer un bulletin météorologique dans plusieurs langues (Wanner et al. 2010)) et ceux de François Lareau et Florie Lambrey, qui ont tous deux également travaillé sur la modélisation de la TST, en étudiant pour cela les collocations dans le cadre des textes multilingues (Lambrey et Lareau 2015) et cherchant à optimiser et améliorer le générateur de texte utilisé dans MARQUIS (Lareau et al. 2018). Si ces travaux concernent surtout les niveaux de représentation sémantique et de syntaxe profonde du modèle sens-texte (MST), la thèse d'Alexis Nasr (Nasr 1996) traite également et surtout des niveaux de représentation morphologique, de syntaxe de surface et de syntaxe profonde, et des modules de transition entre ces niveaux.

En ce qui concerne les ACGs et leur utilisation dans ACGtk, de nombreux travaux ont été réalisés, en particulier ceux de Philippe de Groote sur les ACGs (de Groote 2001), ou ceux de Sylvain Pogodalla sur les grammaires d'arbres adjoints et leur modélisation dans ACGtk à l'aide des ACGs (Pogodalla 2017a ; Pogodalla 2017b). En effet, les grammaires d'arbres adjoints (TAG) sont des formalismes grammaticaux qui manipulent des arbres, et effectuent entre autres des opérations de substitution et d'adjonction (Abeillé 1993), donc des manipulation de structures, et sont utilisées entre autres pour représenter le langage et les phrases. Sylvain Pogodalla a d'ailleurs montré (Pogodalla 2017a) qu'une interface semantique - syntaxe utilisant les TAG était implémentable dans ACGtk, et que cette implémentation permettait la gestion (analyse comme génération) des idiomes tels que « *kicked the bucket* ». Ces résultats sont très intéressants, car je vais être amenée dans ce projet à manipuler des idiomes, et que la TST utilise des structures d'arbres dans ses niveaux de représentation. Je n'aurais certes pas à manipuler de TAG, mais ces travaux m'ont beaucoup inspirée pour ce projet.

# 4 Grammaires Catégorielles abstraites et ACGtk

Les ACGs sont un formalisme grammatical sur lequel mon travail lors de ce stage de fin d'étude se base.

Elles ont l'avantage de permettre l'opération de transduction d'une signature objet (cf. définitions 4.1.2 page suivante et 4.1.6 page suivante) à une autre à condition qu'elles partagent une même signature abstraite. Les structures utilisées par les ACGs et leur fonctionnement semblent particulièrement adapté au fonctionnement de la TST, qui comprend 7 niveaux de représentation, et où les structures de données d'un niveau de sont pas modifiées d'un niveau à un autre, mais où la structure de données du niveau suivant est créée à partir de celle du précédent. Cette utilisation des ACGs fait très fortement écho aux modules de transition de la TST (cf. section 5).

Je décris en sous-section 4.1 ce que sont les ACGs, puis en sous-section 4.2 le logiciel ACGtk et son fonctionnement.

## 4.1 Grammaires Catégorielles Abstraites

Les ACGs sont un formalisme grammatical permettant de coder d'autres formalismes grammaticaux, comme les TAG, ou encore les chaînes de caractères, ainsi que la transformation d'arbres (issus des TAG) en chaînes de caractère (Pogodalla 2017a). Ces grammaires permettent entre autres d'exprimer par les types de $\lambda$-termes les catégories syntaxiques ainsi que les types sémantiques, bien que ce soient deux notions différentes (de Groote 2001).

Les ACGs sont basées sur le $\lambda$-calcul, et utilisent les notions de signatures d'ordre supérieur et de lexiques d'une signature d'ordre supérieur à une autre. Je vais introduire ces notions, puis définir formellement les ACGs, et ensuite expliquer leur utilité.

Un exemple simple et détaillé d'ACG est en annexe C, afin d'illustrer toute cette partie. Les définitions suivantes sont issues de (de Groote 2001).

### 4.1.1 Quelques définitions introductives

**Définition 4.1.1** *Soit $A$ l'ensemble des types atomiques. L'ensemble des* ***types implicatifs linéaires*** *$\mathcal{T}(A)$ est défini sur $A$ par induction :*
- *si $a \in A$ alors $a \in \mathcal{T}(A)$,*
- *si $\alpha, \beta \in \mathcal{T}(A)$ alors $(\alpha \multimap \beta) \in \mathcal{T}(A)$.*

**Définition 4.1.2** *Soit $\Sigma$ une* ***signature d'ordre supérieur****. $\Sigma$ est de la forme $\Sigma = \langle A, C, \tau \rangle$, où $A$ est l'ensemble des types atomiques, $C$ un ensemble de constantes, et $\tau$ une fonction de $C$ dans $\mathcal{T}(A)$ associant aux constantes un type.*

**Définition 4.1.3** *Soit $X$ un ensemble infini dénombrable de $\lambda$-variables. L'ensemble $\Lambda(\Sigma)$ des* ***$\lambda$-termes linéaires*** *est défini sur $\Sigma$ par induction :*
- *si $c \in C$ alors $c \in \Lambda(\Sigma)$,*
- *si $x \in X$ alors $x \in \Lambda(\Sigma)$,*
- *si $x \in X$, $t \in \Lambda(\Sigma)$ et si $x$ n'apparait libre dans $t$ qu'une seule fois (linéarité), alors $(\lambda x.t) \in \Lambda(\Sigma)$,*
- *si $t, u \in \Lambda(\Sigma)$ et si les ensembles des variables libres de $t$ et $u$ sont disjoints alors $(tu) \in \Lambda(\Sigma)$.*

Les propriétés du $\lambda$-calcul de $\alpha$-réduction et $\beta$-réduction sont conservées pour $\Lambda(\Sigma)$. En outre, les règles logiques de modus ponens, introduction de l'implication et l'axiome (de Groote 2001) s'adaptent à $\Lambda(\Sigma)$, cf. section C.2. On peut donc dériver et prouver que des termes de $\Lambda(\Sigma)$ soient bien typés. Si un terme $t \in \Lambda(\Sigma)$ de de type $\alpha$ est bien typé, on utilisera la notation $\vdash_\Sigma t : \alpha$.

### 4.1.2 Les ACGs

**Définition 4.1.4** *Soient deux signatures (ou vocabulaires) $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ et $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$. Le* ***lexique*** *$\mathcal{L} = \langle F, G \rangle$ de $\Sigma_1$ dans $\Sigma_2$ est un tuple tel que :*
- *$F : \mathcal{T}(A_1) \to \mathcal{T}(A_2)$ est un homomorphisme,*
- *$G : \Lambda(\Sigma_1) \to \Lambda(\Sigma_2)$ est un homomorphisme,*
- *$F$ et $G$ sont tels que $\forall c \in C_1 \vdash_{\Sigma_2} G(c) : F(\tau_1(c))$ est prouvable.*

**Définition 4.1.5** *Une* ***Grammaire catégorielle abstraite*** *est un tuple $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ où :*
- *$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ et $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ sont deux signatures d'ordre supérieur,*
- *$\mathcal{L} = \Sigma_1 \to \Sigma_2$ est le lexique,*
- *$s \in \mathcal{T}(A_1)$ est le type distingué de la grammaire.*

**Définition 4.1.6** *Le* ***langage abstrait*** *$\mathcal{A}$ et le* ***langage objet*** *$\mathcal{O}$ d'une ACG $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ sont :*

$$\mathcal{A} = \{t \in \Lambda(\Sigma_1) | \vdash_{\Sigma_1} t : s \text{ est dérivable}\}$$
$$\mathcal{O} = \{t \in \Lambda(\Sigma_2) | \exists u \in \mathcal{A}(\mathcal{G}) \text{ tel que } t = \mathcal{L}(u)\}$$

*Le langage abstrait d'une ACG est l'ensemble des structures grammaticales abstraites d'un type donné. Le langage objet d'une ACG est l'ensemble des réalisations concrètes générées par le langage abstrait.*

On parle d'interprétation (cf. figure 1 page ci-contre) lorsqu'on calcule, pour un terme $u$ du langage abstrait, l'élément correspondant $\mathcal{L}(u)$ dans le langage objet. Réciproquement, on parle d'analyse (ou de parsing) (cf. figure 1 page suivante) lorsque pour un terme du langage objet $t$, on cherche tous les antécédents $u$ possibles tels que $\mathcal{L}(u) = t$. Comme évoqué précédemment, on peut composer des ACGs. En particulier, si $\Lambda(\Sigma_1)$ est le langage abstrait de deux ACGs différentes $\mathcal{G}_{12}$ et $\mathcal{G}_{13}$, il est en correspondance avec deux vocabulaires objets différents $\Sigma_2$ et $\Sigma_3$ (cf. figure 1 page ci-contre). L'opération $T$ de transduction entre $\Sigma_2$ et $\Sigma_3$ est telle que $T(a, b) \Leftrightarrow \exists t. \mathcal{L}_{12}(t) = a \wedge \mathcal{L}_{13}(t) = b$, où $t \in \Sigma_1$, $a \in \Sigma_2$ et $b \in \Sigma_3$.

**Définition 4.1.7 (Pogodalla 2017b)** *L'**ordre** d'une ACG est le maximum de l'ordre de ses constantes abstraites. L'ordre d'une constante abstraite est l'ordre de son type $\tau$. L'ordre d'un type $\tau \in \mathcal{T}(A)$ est défini par induction :*

— $ordre(\tau) = 1$ *si* $\tau \in A$,

— $order(\alpha \multimap \beta) = ordre(\alpha \to \beta) = max(1 + ordre(\alpha), ordre(\beta))$ *sinon.*

*La **complexité** d'une ACG est le maximum des ordres des réalisations de ses types atomiques. Une ACG d'ordre $\gamma$ et de complexité $\eta$ est notée $ACG_{(\gamma,\eta)}$.*

### 4.1.3 Applications des ACGs

Les ACGs ont trois applications principales (de Groote 2001), à savoir les applications applicative, déductives et transductive. L'application applicative indique que l'on peut calculer selon le lexique d'une signature l'image dans le langage objet d'un terme abstrait. L'application déductive permet de décider si un terme donné appartient au langage objet d'une ACG. L'application transductive consiste en la composition des application applicatives et déductives. Elle permet le transfert d'un langage objet à un autre langage objet, générés tous deux à partir du même langage abstrait (voir figure 1).

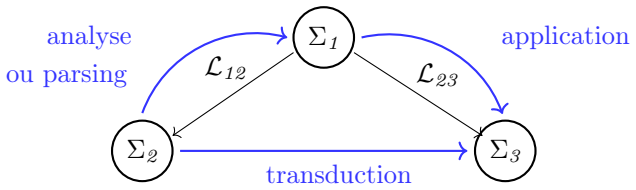



FIGURE 1 – Schéma illustratif d'un ensemble d'ACGs

En TALN, la génération de texte consiste à partir d'une représentation (souvent sémantique) de ce texte, pour produire le texte en question. Cela va se traduire dans les ACGs en l'enchaînement de plusieurs ACGs. En effet, la relation sens-texte décrite dans la TST peut être encodée comme une composition de fonctions et d'inverse de fonctions (i.e. comme une composition de transductions). Cela permet alors d'encoder un lien relationnel par une composition de liens fonctionnels. Ainsi, pour générer du texte à partir d'une représentation sémantique jusqu'à une représentation de syntaxe de surface par exemple, il suffit donc de sélectionner le terme voulu dans la signature correspondant à cette représentation sémantique, et d'enchaîner les opérations d'analyse, d'application ou de transduction jusqu'à la signature de représentation de syntaxe de surface.

Ces trois applications permettent de générer un certain nombre de langages, avec une complexité au niveau de l'application déductive (parsing) polynomiale dans de très nombreux cas. En effet, si l'ordre d'une ACG est inférieur ou égal à 2, cette complexité est polynomiale. La figure 2 (Pogodalla 2017b) illustre le pouvoir expressif des ACGs (en ce qui concerne les langages de chaînes de caractères) :

| ACG | langage généré |
|---|---|
| $ACG(1, n)$ | langages finis |
| $ACG(2, 1)$ | langages réguliers |
| $ACG(2, 2)$ | langages hors-contexte |
| $ACG(2, 3)$ | grammaires hors-contexte multiples bien équilibrées |
| $ACG(2, 4)$ | grammaires faiblement contextuelles |
| $ACG(2, 4 + n)$ | $ACG(2, 4)$ |
| $ACG(3, n)$ | décidabilité de MELL |

FIGURE 2 – Pouvoir expressif des ACGs

## 4.2 ACGtk

ACGtk est un logiciel développé en OCaml par l'équipe Sémagramme, en particulier par Sylvain Pogodalla depuis 2008, et maintenu et amélioré également par Vincent Tourneur, ingénieur de recherche dans l'équipe Sémagramme. Différents membres de l'équipe Sémagramme participent également à son développement. En effet, le code source est sur un dépôt Gitlab, ce qui permet un bon suivi du développement et une mise à jour facile du logiciel lorsqu'une nouvelle version est publiée.

ACGtk est un logiciel pour le développement des ACGs. Il est constitué de deux executables, `acg` et `acgc`. Similairement au fonctionnement des langages de programmation que l'on compile puis exécute, une ACG (encodée dans un fichier `.acg`) est compilée (par `acgc`) en un analyseur (encodé dans un fichier `.acgo`, généré par ll'executable `acgc`), analyseur que l'on peut utiliser avec le langage de commandes d'`acg`.

Vous pourrez trouver des exemples d'utilisation d'ACGtk dans la section 7.

# 5 Théorie Sens-Texte

Mon stage a consisté en l'implémentation d'une architecture de plusieurs ACGs, afin de modéliser le fonctionnement de la TST. Même si je présente dans ce rapport les sept niveaux de représentation utilisés dans la TST, je n'ai modélisé que les trois premiers, à savoir les niveaux de représentation sémantique, syntaxique profonde et syntaxique de surface. C'est pourquoi cette section ne détaille que ceux-ci. Des informations complémentaires et une brève présentation des autres niveaux de représentation se trouvent en annexe A. La TST s'intéresse à la représentation des sens et des textes, et cherche à répondre à la question : comment est-ce qu'un sens **S** est exprimé dans le langage **L** ?

Les 4 sous-sections suivantes présentent brièvement cette théorie linguistique, qui est plus amplement détaillée en annexe A.

## 5.1 Présentation générale

Un modèle sens-texte (MST) est composée de 7 niveaux de représentation (Milićević 2006) interdépendants (cf. figure 3).

SemR ↔ DSyntR ↔ SSyntR ↔ DMorphR ↔ SMorphR ↔ DPhonR ↔ SPhonR

FIGURE 3 – Les septs niveaux de représentation de la TST

Ces sept niveaux permettent l'analyse d'un texte (sens $Texte \rightarrow Sens$) comme la synthèse d'un sens (sens $Sens \rightarrow Texte$) (Mel'čuk et al. 2012). Nous nous intéressons ici particulièrement à la direction $Sens \rightarrow Texte$. Le niveau de représentation **SemR** est celui le plus proche du sens, et le niveau de représentation **SPhonR** celui le plus proche du texte.

**Définition 5.1.1 (Milićević 2006)** *Le **sens** d'un énoncé est le contenu linguistic (de cet énoncé) à communiquer (i.e. quelque chose d'intelligible et de traductible). Le sens d'un énoncé est composé d'éléments de trois types (Nasr 1996) :*

- *le sens situationnel, qui représente l'état de la chose, ce qui est désigné (i.e. un objet, un évènement, des relations entre objets, etc.)*
- *le sens communicatif, qui représente l'organisation du message par le locuteur, et désigne quelle partie du discours ou de la phrase sera mis en exergue.*
- *le sens rhétorique, qui représente les effects expressifs ou artistiques voulus par le locuteur*

**Définition 5.1.2 (Milićević 2006)** *Le **texte** d'un énoncé est un fragment de discours (i.e. quelque chose d'immédiatement perceptible). Il peut être d'une longueur quelconque.*

**Exemple 5.1.3** ***Analogie avec un verre d'eau :*** *Si on veut donner de l'eau à quelqu'un, on va lui donner un verre d'eau, ou une bouteille d'eau, mais pas l'eau directement. En effet, on ne peut pas attraper de l'eau, il nous faut un contenant. On peut faire un analogie avec le sens et le texte : l'eau est le sens et le verre le texte. En effet, si on veut dire à quelqu'un qu'on l'aime, on peut dire* « Je t'aime. » *ou bien* « Je suis tombé amoureux de toi. »*. Dans les deux cas, on aura transmis la même chose, quelque chose d'intangible, au moyen de texte, immédiatement perceptible.*

On peut de plus parler d'observabilité. En effet, un texte est immédiatement perceptible (et donc observable également) et permet donc d'observer les sens, qui ne sont, eux, pas immédiatement perceptibles. Cependant, lorsqu'une personne lit ou entend deux phrases, elle est capable de dire si ce sont des paraphrases exactes (comme « *il est grand* » et « *il est de grande taille* »), des paraphrases approximatives (comme « *L'arrivée tardive de John a étonné Mary.* » et « *Le fait que John soit arrivé tard a étonné Mary.* » (Mel'čuk et al. 2013)), ou des phrases qui n'ont aucun lien de paraphrase (comme « *La*

*banane est bleue.* » et « *Il fait beau dehors.* »). On peut donc observer et comparer des sens grâce aux textes, car la paraphrase est bel et bien une notion de sens et non de texte (cf. definition 5.1.4).

**Définition 5.1.4 (Polguère 2003)** *Deux expressions linguistiques ayant (à peu près) le même sens sont appelées des* ***paraphrases****.*

La TST est basée sur trois postulats. Le premier indique que le langage est considéré comme une correspondance multiple entre un ensemble infini dénombrable de sens, et ensemble infini dénombrable de textes, comme je l'expliquais plus haut à propos des représentations des sens et des textes. Le second postulat introduit le modèle sens-texte (MST). En effet, il explique que cette correspondance sens-texte est décrite par un outil formel qui simule l'activité langagière d'un locuteur natif. Cet outils formel est le MST. Le troisième postulat introduit lui les niveaux intermédiaires (entre le sens, i.e. niveau sémantique, et le texte, i.e. niveau phonologique). Ainsi, il indique qu'étant donnée la complexité de la correspondance sens-texte, des niveaux de représentation intermédiaires sont nécessaires, à savoir les niveaux syntaxiques et morphologiques.

Un modèle sens-texte est donc constitué de ces sept niveaux de représentation ainsi que de modules de transition pour passer d'un niveau à un autre (cf. figure 25 page 45), que ce soit pour générer du texte (synthèse) ou pour comprendre un sens (analyse). Lors de ces transitions, la notion de paraphrase est beaucoup utilisée, afin de définir un sens, ou de générer une phrase ayant un sens donné.

De plus, les septs niveaux de représentation de la TST utilisent les notions de sémantique (cf. définition 3.1.1 page 2), syntaxe (cf. définition 3.1.3 page 2), morphologie (cf. définition B.0.2 page 49) et phonétique (cf. définition A.3.1 page 47).

## 5.2 Les 7 niveaux de représentation

Chaque niveau de représentation possède plusieurs composantes, dont une principale. Nous allons ici décrire ces composantes et leur structure. Vous trouverez plus de détails sur ces niveaux en annexe A.

### 5.2.1 SemR : la représentation sémantique

La représentation sémantique est composée de 4 structures. Sa structure principale est la structure sémantique **SemS**, et les trois autres sont les structures sémantique communicative **Sem-CommS**, rhétorique **RhetS**, et la structure de référence **RefS** (cf. figure 4).

$$\textbf{SemR} = \langle \text{ SemS, Sem-CommS, RhetS, RefS } \rangle$$

FIGURE 4 – Composantes de la **SemR**

La structure sémantique **SemS** est représentée sous la forme d'un graphe connexe (Milićević 2006). Les nœuds de ce graphe sont des sémantèmes et ses arcs sont étiquetés par des nombres (différents) qui indiquent la relation entre le prédicat (le sémantème) et ses arguments (cf. figure 5).

**Définition 5.2.1 (Nasr 1996)** *Un* ***sémantème*** *est une unité sémantique, qui représente sens d'un lexème ou d'un phrasème (cf. définition 5.2.3 page suivante).*

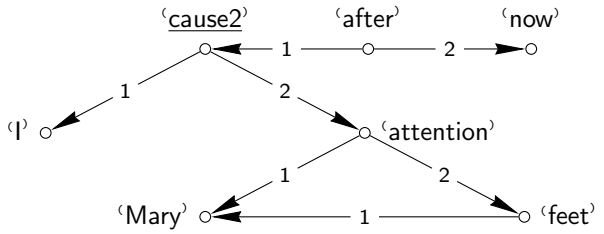



FIGURE 5 – Exemple de structure sémantique (Mel'čuk et al. 2012, p106) pour la phrase « *I will draw Mary's attention to her feet.* »

### 5.2.2 DSyntR : la représentation syntaxique profonde

La représentation syntaxique profonde est composée de 4 structures. Sa structure principale est la structure syntaxique profonde **DSyntS**, et les trois autres sont la structure syntaxique profonde communicative **DSynt-CommS**, la structure syntaxique profonde prosodique **DSynt-ProsS**, et la structure syntaxique profonde anaphorique **DSynt-AnaphS**, (cf.figure 6).

**DSyntR** = ⟨ **DSyntS**, **DSynt-CommS**, **DSynt-ProsS**, **DSynt-AnaphS** ⟩

FIGURE 6 – Composantes de la **DSyntR**

La structure syntaxique profonde **DSyntS** est représentée sous la forme d'un arbre de dépendances non ordonné (Milićević 2006). Les nœuds de cet arbre sont des lexèmes profonds, et les branches sont étiquetées par les relations syntaxiques profondes (cf. figure 7).

**Remarque 5.2.2** *L'ordre des mots n'est ici pas représenté, seules les dépendances entre eux le sont, et les lexèmes vides (tels que les auxiliaires, les prépositions) ainsi que les pronoms et mots pronominaux ne sont pas représentés ici. De plus, les phrasèmes seront représentés par un seul nœud ici (Nasr 1996) (cela changera au niveau de la représentation syntaxique de surface).*

**Définition 5.2.3 (Polguère 2003)** *Un **lexème** est une UL, c'est-à-dire une entité générale qui se manifeste ou matérialise dans les phrases par des mots formes. Un **phrasème** est un groupement de lexèmes qui représente un sens autre que les sens individuels de ses composants (par exemple* « tout à coup » *sera représenté par un phrasème).*

**Définition 5.2.4** *Les **relations syntaxiques profondes** sont des relations de dépendances entre un gouverneur (le nœud de "départ") et un dépendant (le nœud d'"arrivée"). Elles sont de plus asémantiques. Les relations syntaxiques profondes sont au nombre de 12, et peuvent être des relations de coordination (telles que COORD et QUASI-COORD), ou des relations de subordination. Dans ces dernières, on retrouve la relation APPEND, les relations attributives ATTR et $ATTR_{descr}$, et les relations actancielles I, II, $II_{dir-sp}$, III, IV, V et VI.*

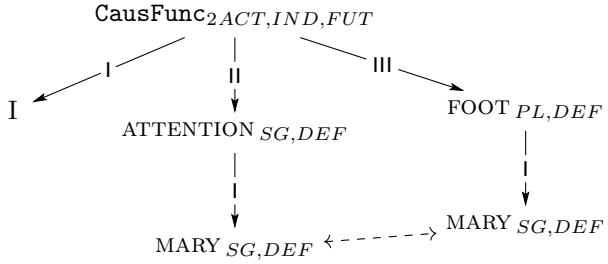



FIGURE 7 – Exemple de structure syntaxique profonde (Mel'čuk et al. 2012, p107) pour la phrase « *I will draw Mary's attention to her feet.* » (la flèche en pointillés fait partie de la structure anaphorique cf. annexe A).

### 5.2.3 SSyntR : la représentation syntaxique de surface

La représentation syntaxique de surface est composée de 4 structures. Sa structure principale est la structure syntaxique de surface **SSyntS**, et les trois autres sont les structures syntaxique de surface communicative **SSynt-CommS**, syntaxique de surface prosodique **SSynt-ProsS**, et syntaxique de surface anaphorique **SSynt-AnaphS** (cf.figure 8).

**SSyntR** = ⟨ **SSyntS**, **SSynt-CommS**, **SSynt-ProsS**, **SSynt-AnaphS** ⟩

FIGURE 8 – Composantes de la **SSyntR**

La structure syntaxique de surface **SSyntS** est représentée sous la forme d'un arbre de dépendances non ordonné, comme **DSyntS** (Milićević 2006). Les nœuds de cet arbre sont les lexèmes effectifs de la

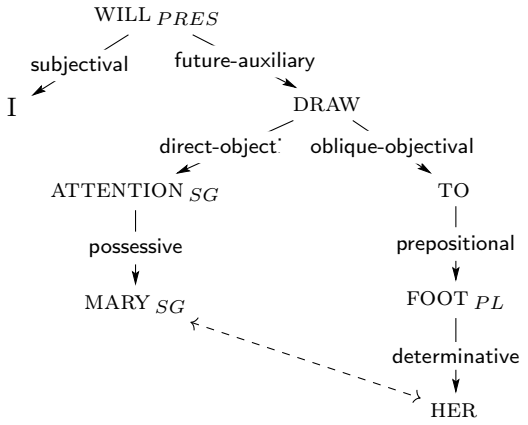



FIGURE 9 – Exemple de structure syntaxique de surface (Mel'čuk et al. 2012, p109) pour la phrase « *I will draw Mary's attention to her feet.* ».

phrase, et les branches sont étiquetées par les noms des relations de dépendances syntaxiques de surface spécifiques au langage (cf. figure 9).

**Remarque 5.2.5** *Alors que dans la* ***DSyntS*** *les lexèmes étaient sémantiquement complets, ce sont ici les lexèmes qui seront présents dans la phrase, et qui peuvent être asémantiques parfois, comme c'est le cas des verbes supports. Par exemple, dans la phrase* « I will draw Mary's attention to her feet », DRAW *prend deux actants syntaxique de surface* ATTENTION *et* TO *(cf. figure 9), alors qu'en syntaxe profonde* `CausFunc` *est utilisée (cf. figure 7 page ci-contre), où* `CausFunc` *est la FL indiquant le besoin d'un verbe support. En effet, dans* « draw attention », DRAW *n'a "pas de sens". On ne dessine pas de l'attention, on est dans un état d'éveil.* DRAW *sert ici à exprimer cela, sans toutefois avoir de sens. C'est un verbe support.*

*Aussi, la* ***DSyntS*** *donnait des relations de dépendance universelles alors que la* ***SSyntS*** *donne des relations de dépendances spécifiques, et les lexèmes vides (tels que les auxiliaires, les prépositions) ainsi que les pronoms et mots pronominaux sont maintenant représentés. De plus, les phrasèmes seront représentés par plusieurs nœuds (Nasr 1996). Par exemple,* « appuyer sur le champignon » *sera représentée en syntaxe profonde par un seul lexème (par exemple* `syn`*(*ACCÉLÉRER *), où* `syn` *est la FL de synonymie), et en syntaxe de surface par trois lexèmes de surface,* APPUYER , SUR , CHAMPIGNON *, reliés par les bonnes relations de syntaxe de surface (*complément d'objet direct *puis* préposition*, dans l'ordre).*

## 5.3 Les modules de transition

Les passages d'un niveau de transition à un autre sont effectués par des modules de transition, qui appliquent des règles (propres à chaque module) afin de passer d'une représentation à une autre. Chaque module prend le nom du niveau de représentation le plus bas (le niveau le plus proche du sens, donc de la représentation sémantique) des deux entre lesquels il est. Le lexique utilisé pour ces transitions est le dictionnaire explicatif combinatoire (DEC). Nous allons ici décrire ces modules.

Comme indiqué précédemment, je n'ai travaillé que sur les niveaux de représentation sémantique, syntaxique profond, et syntaxique de surface. Aussi, je n'ai travaillé que sur les modules sémantiques et de syntaxe profonde. Je parle du module syntaxique de surface en annexe A.

### 5.3.1 Le module sémantique : SemR ↔ DSyntR

Ce module permet la transition de la représentation semantique à la représentation syntaxique profonde (Mel'čuk et al. 2013). Il est constitué de trois étapes : l'étape de paraphrase sémantique, la transition de la **SemR** à la **DSyntR**, et l'étape de paraphrase syntaxique profonde.

Concernant les paraphrases sémantiques, deux types de paraphrases existent :

— Les paraphrases linguistiques, qui n'ont pas besoin de connaissance extérieure (sur le monde, ou des connaissances arithmétiques par exemple).

— Les paraphrases extra-linguistiques, qui utilisent la culture générale et d'autres connaissances. Par exemple, « *Le plus vaste pays de la planète* » et « *la Russie* » sont des paraphrases mais n'ont pas le même sens à proprement parler. Pour établir un tel lien de synonymie on a besoin de connaissance autre que purement sémantique sur le monde.

Dans l'étape de paraphrase sémantique, seule la paraphrase linguistique est prise en compte. Il s'agit principalement de définir un sémantème à l'aide d'autres sémantèmes ayant des sens plus simples. Ces définitions sont données dans le dictionnaire explicatif combinatoire (cf. section 5.4), qui est le dictionnaire utilisé par la TST.

Lors de la transition de la représentation sémantique vers la représentation syntaxique profonde, des règles de transition sémantique sont utilisées, réparties en quatre types principaux :

— Les règles sémantiques lexicales syntaxiques : elles font la transition entre **SemS** et (**DSyntS** & **DSynt-AnaphS**). Ce sont le type de règles le plus important de la transition, et sont composés des règles d'arborisation (pour construire l'arbre de dépendance de la **DSyntS** à partir de la **SemS**), et des règles de lexicalisation (qui vont associer à chaque sémantème un lexème).
— Les règles sémantiques prosodiques : elles font la transition entre **SemS** et **DSynt-ProsS**.
— Les règles sémantiques communicatives : elles font la transition entre **Sem-CommS** et (**DSyntS** & **DSynt-CommS**).
— Les règles sémantiques communicatives : elles font la transition entre **RhetS** et (**DSyntS** & **DSynt-ProsS**).

Aussi, les deux opérations principales de cette étapes sont les opérations de lexicalisation sur la **SemS** puis d'arborisation, afin d'obtenir la **DSyntS** correspondante (Milićević 2006). L'opération de lexicalisation permet de faire correspondre les sémantèmes (utilisés dans la **SemS**) à leurs lexèmes (qui seront utilisés dans la **DSyntS**).

Les règles de lexicalisation sont des règles nodales. C'est-à-dire qu'elles spécifient les étiquettes des nœuds de la **DSyntS**, et tout ce qui y est relié. Certaines d'entre elles produisent par exemple des lexèmes ou des idiomes, et d'autres ont affaires aux FLs.

Les règles d'arborisation sont des règles concernant les arêtes de la **DSyntS**. En effet, elles spécifient le squelette de celle-ci, et déterminent son nœud dominant ainsi que les relations étiquetées sur les branches de la **DSyntS**.

L'étape de paraphrase syntaxique profonde est basée sur les FLs, et composée de deux principaux types de règles (cf. exemple 6.4.1 page 24 et un exemple d'application de ces règles en annexe D) :

— Les règles syntaxiques profondes lexicales, qui s'occupent des équivalences sémantiques (par exemple, « *être calme* » est équivalent à « *ne pas être agité* »),
— Les règles syntaxiques profondes de restructuration, qui s'occupent des changements syntaxiques et des restructurations de l'arbre, afin de permettre aux règles syntaxiques profondes lexicales de s'appliquer. En effet, un changement de lexème peut entrainer des changements de structure afin de préserver le sens initial. L'annexe D illustre cela lors de l'ajout de `non` dans l'arbre lors de l'utilisation de la règle lexicale 24 (cf. exemple 6.4.1 page 24).

### 5.3.2 Le module syntaxique profond : DSyntR ↔ SSyntR

Ce module permet la transition de la représentation syntaxique profonde à la représentation syntaxique de surface (Milićević 2006). Il est constitué de six types de règles principalement ; je me suis occupée des trois premiers types de règles :

— Les règles phrasémiques étendent les nœuds de la **DSyntS** qui étaient des phrasèmes en des sous-arbres de la **SSyntS**, dont les nœuds sont cette fois des lexèmes, en calculant les valeurs des FLs présentes.
— Les règles syntaxiques profondes construisent la **SSyntS** à partir de la **DSyntS** selon les dépendances syntaxiques profondes et les propriétés lexicographiques des lexèmes, selon le patron de gouvernement (spécifié dans le DEC).
— Les règles de pronominalisation introduisent les pronoms "manquants" et ceux correspondant aux liens de co-références (de la DSyntR).
— Les règles d'ellipses permettent les ellipses, en fonction des règles de la langue en question.
— Les règles communicatives construisent la **SSynt-CommS** à partir de la **DSynt-CommS**.
— Les règles prosodiques construisent la **SSynt-ProsS** à partir de la **DSyntR**.

## 5.4 Le Dictionnaire Explicatif Combinatoire

Le Dictionnaire Explicatif Combinatoire (DEC) est le lexique utilisé par la TST. En effet, les règles présentes dans les différents modules viennent d'informations lexicales, il faut donc un dictionnaire pour regrouper ces règles, c'est le DEC. Il intervient entre autres au niveau de la paraphrase sémantique, pour le choix des FLs, pour le choix des relations syntaxiques de surface en fonction des relations syntaxiques profondes. En effet, chaque entrée lexicale y possède une définition (cf. exemple 5.4.2), celle utilisée pour la paraphrase sémantique (cf. section 5.3.1).

**Remarque 5.4.1** *Les entrées lexicales du DEC sont différentes pour chaque sens différent, même si un même mot (qui est alors une super-entrée) peut avoir différents sens. Dans un dictionnaire classique, on aura une énumération de ces sens sous le mot en question. De manière similaire, ici ces variantes seront indiquées sous la super-entrée de manière résumée, avec une courte définition et un exemple, puis de manière détaillée une à une. Ces sous-entrées sont distinguées par un numéro accolé à l'unité lexicale (cf. exemple 5.4.2).*

**Exemple 5.4.2 (Mel'čuk et al. 2013)** *Définition sémantique pour l'entrée sémantique* X bakes Y in Z *:* X bakes Y in Z *: X cooks***1** *solid***1** *Y by submitting Y to the indirect***1** *action***6** *of dry***10** *heat***2** *in device***1** *Z or in contact***1** *with source***1** *Z of heat***2**.

Cette définition se veut "plus simple" [1] que son entrée, voire irréductible. Idéalement, cette définition est irréductible : on ne peut pas "déplier" cette définition pour obtenir une définition plus précise.

De plus, le patron de gouvernement, ou *Government Pattern* est aussi précisé pour chaque entrée (cf. exemple 5.4.3). C'est lui qui est utilisé pour convertir les relations sémantiques en relations syntaxiques profondes, puis pour convertir ces relations syntaxiques profondes en relation syntaxique de surface. Il indiquera aussi les prépositions à rajouter au niveau de la représentation syntaxique de surface.

**Exemple 5.4.3 (Mel'čuk et al. 2013)** *Government Pattern de l'entrée* X bakes Y in Z *du DEC :*

| $X \Leftrightarrow$ **I** | $Y \Leftrightarrow$ **II** | $Z \Leftrightarrow$ **III** |
|---|---|---|
| *1.N* | *1.N* ***obligatory*** | $1.PREP_{loc}N$ |

On y retrouve également pour chaque entrée une liste exhaustive de toutes les FLs pouvant s'appliquer à cette entrée (cf. exemple 5.4.4), et les valeurs qu'elles peuvent prendre, avec des exemples.

**Exemple 5.4.4** *[Mel'čuk et al. 2013] Extrait de la liste des FLs pouvant s'appliquer à l'entrée* X bakes Y in Z *du DEC :*

| | |
|---|---|
| `Syn` | *:* make baked Y |
| `Gener` | *:* cook**1** |
| $\mathtt{S}_{instr-loc}$ | *:* baking tray/sheet ; baking tin ; cake tin ; foil ; tinfoil |

Enfin, des remarques ou indications peuvent aussi être présentes, afin de mieux utiliser les données mentionnées ci-dessus.

Le DEC a pour but d'aider le locuteur à trouver comment exprimer le sens qu'il cherche à exprimer, et c'est pour cela qu'il est utilisé dans la TST. Une ressource lexicale similaire au DEC est actuellement en cours de développement pour le français, il s'agit du RLF (Réseau lexical du Français).

# 6 Solutions proposées

Le but de ce projet est de générer du texte à l'aide des ACGs et de la TST. Je me suis concentrée sur les niveaux de représentation sémantique, syntaxique profonde et syntaxique de surface de la TST.

On a vu précédemment que la TST est décrite par un MST composé de 7 niveaux de représentation. Comme je ne traite que les 3 premiers, je vais maintenant parler de ceux-ci. Chacun de ces niveaux a une structure de données associée, afin de représenter l'une des étapes de la transition sens-texte

1. Par "plus simple" je veux dire une définition qui se veut claire, précise, et qui va empêcher les boucles de définitions qui se renvoient l'une à l'autre. Aussi, les sémantèmes utilisés dans cette définition se voudront "plus simples" que celui qui est défini, car leur définition ne pourra jamais renvoyer au sémantème défini initialement.

d'une phrase. Ici, deux modules de transition interviennent donc (le module sémantique et le module syntaxique profond). Un module de transition crée la structure du niveau de représentation suivant à partir de celle du niveau de représentation précédent. Cette opération est similaire de l'opération de transduction permise par les ACGs, qui permet de passer du langage objet d'une ACG au langage objet d'une autre ACG, tant que ces deux ACGs possèdent le même langage abstrait. Cette opération peut s'effectuer dans les deux sens, ce qui est en accord avec la TST qui peut générer comme analyser du texte. Intuitivement, cela semble donc naturel de représenter chaque niveau de représentation de la TST par un langage objet, et de trouver des signatures communes à ces langages objets afin de pouvoir appliquer l'opération de transduction de l'un à l'autre. Cependant, cela s'est révélé plus compliqué que l'intuition, et d'autres opérations de transduction ont dû être insérées (cf. figure 10 page suivante).

Il s'agit donc d'écrire des ACGs afin de représenter un ensemble restreint mais complet de phrases pour les trois niveaux de représentation en question. Par ensemble restreint et complet, j'entend un ensemble relativement petit de quelques phrases, mais qui illustre bien la plupart des phénomènes que l'on peut rencontrer dans la langue naturelle (comme les collocations ou expressions idiomatiques par exemple) ainsi que les phénomènes ayant lieu de par la construction de la TST (comme la paraphrase sémantique, la paraphrase syntaxique profonde ou les fonctions lexicales par exemple).

Cette section détaille le modèle de génération que j'ai implémenté, avec en sous-section 6.1 un aperçu général de l'articulation de ce modèle, en sous-section 6.2 les principes de représentation généraux que j'ai utilisé et les choix généraux d'implémentation que j'ai été amenée à faire et pourquoi, puis en sous-sections 6.3, 6.4 et 6.5 les détails concernant respectivement les niveaux sémantiques, syntaxique profond et syntaxique de surface de cette implémentation.

Des figures contenant des extraits de code afin d'illustrer les sections suivantes sont utilisées, elles se trouvent en annexe H.

## 6.1 Aperçu de l'implémentation réalisée

Au cours de ce stage j'ai donc implémenté un modèle pour la TST dans ACGtk. Les parties suivantes détailleront mes réflexions et solutions théoriques, je donne ici un aperçu de la solution pratique au problème posée, pour une meilleure compréhension de ce qui suit.

Mon implementation est constituée de 9 signatures et 8 lexiques, divisibles en 5 zones (voir illustration en figure 10 page ci-contre) :

- la zone sémantique (en rouge), constituée de la signature $\Sigma_{semantic}$, qui correspond à la représentation sémantique, et au lexique $\mathcal{L}_{sem}$,
- la zone de syntaxe profonde (en bleu), constituée des signatures $\Sigma_{deep\text{-}syntactic}$ et $\Sigma_{deep\text{-}rel}$ (cette dernière correspond au niveau de représentation de syntaxe profonde), et du lexique $\mathcal{L}_{dsyntRel}$ reliant les deux signatures précédentes,
- la zone de paraphrase syntaxique (en vert), constituée des deux signatures $\Sigma_{comp}$ et $\Sigma_{lexeq}$ et des lexiques $\mathcal{L}_{compRel}$ et $\mathcal{L}_{compEq}$ ,
- la zone gérant les FLs (en violet), constituée des signatures $\Sigma_{fl}$ et $\Sigma_{finfl}$, et des lexiques $\mathcal{L}_{lexfl}$ et $\mathcal{L}_{reducefl}$,
- la zone de syntaxe de surface (en jaune-orangé), constituée des signatures $\Sigma_{surface\text{-}syntactic}$ et $\Sigma_{surface\text{-}tree}$ (cette dernière correspond au niveau de représentation de syntaxe de surface), et des lexiques $\mathcal{L}_{ssynt}$ et $\mathcal{L}_{stree}$.

On peut composer à chaque fois ces ACGs afin de faire appel à l'opération de transduction, notamment (cf. figure 10 page suivante) :

(a) entre $\Sigma_{semantic}$ et $\Sigma_{deep\text{-}rel}$,
(b) entre $\Sigma_{deep\text{-}rel}$ et $\Sigma_{lexeq}$,
(c) entre $\Sigma_{deep\text{-}rel}$ et $\Sigma_{finfl}$,
(d) et entre $\Sigma_{finfl}$ et $\Sigma_{surface\text{-}syntactic}$.

En effet, ces paires de signatures sont à chaque fois le langage objet de deux ACGs partageant le même langage abstrait (cf. section 4.1).

Pour générer du texte, on part du niveau de représentation sémantique pour aller vers le niveau de représentation de syntaxe de surface. Pour cela, il faut parcourir les signatures du graphe de la figure 10 page ci-contre dans l'ordre suivant :

- (*Niveau sémantique*) $\Sigma_{semantic}$, $\Sigma_{deep\text{-}syntactic}$, $\Sigma_{deep\text{-}rel}$ (*premier arbre syntaxique profond*)
- (*Apparition des FLs potentielles*) $\Sigma_{fl}$, $\Sigma_{finfl}$, $\Sigma_{fl}$, $\Sigma_{deep\text{-}rel}$,

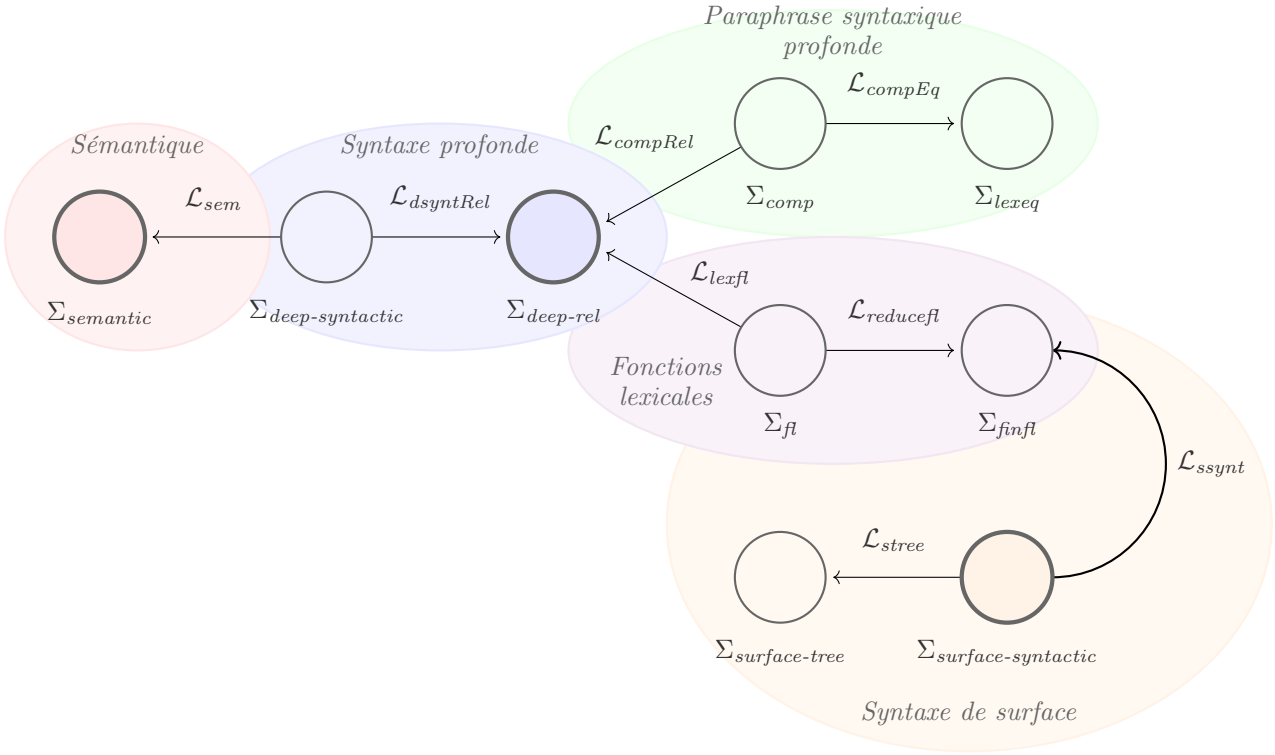



FIGURE 10 – Apercu de l'architecture des ACGs

— (*paraphrase syntaxique profonde*) ($\Sigma_{comp}$, $\Sigma_{lexeq}$, $\Sigma_{comp}$ et $\Sigma_{deep\text{-}rel}$) autant de fois que nécessaire (cf. section 6.4.3),
— (*réalisation des FLs*) $\Sigma_{fl}$, $\Sigma_{finfl}$,
— $\Sigma_{surface\text{-}syntactic}$, $\Sigma_{surface\text{-}tree}$ (*représentation de syntaxe de surface*)

On peut voir dans cette modélisation un parallèle entre les systèmes génératifs des ACGs (les signatures) et de la TST (les niveaux de représentation). En effet, certaines signatures sont l'équivalent des niveaux de représentation de la TST : $\Sigma_{semantic}$ correspond au niveau de représentation sémantique, $\Sigma_{deep\text{-}rel}$ correspond au niveau de représentation syntaxique profond, et $\Sigma_{surface\text{-}tree}$ correspond au niveau de représentation de syntaxique surface. De même, il y a un parallèle entre les modules de la TST et les transductions de cette implémentation. Ainsi, l'enchaînement des transductions (a), (c), (c-inversée), (b), (b-inversée) correspond au module sémantique. L'enchaînement des transductions (c) et (d) correspond au module de syntaxe profonde.

## 6.2 Principes généraux mis en oeuvre

**Choix des ACGs et transduction :** J'ai modélisé les trois niveaux sémantique, de syntaxe profonde et de syntaxe de surface de la TST, en exploitant le phénomène de transduction permis par les ACGs. En effet, comme la TST utilise, pour chaque niveau de représentation, des structures de données construites à partir de celles du niveau précédent, sans modifier la structure précédente, cette propriété est très utile. Cette opération de transduction a en outre l'avantage de pouvoir faire pointer un ensemble de termes d'une signature abstraite sur un seul terme d'un langage objet (je trouve l'image d'un entonnoir parlante pour illustrer ce phénomène), et ce pour deux langages objets différents ayant le même langage abstrait. Autrement dit, au départ on aura une constante, à l'arrivée également une constante, mais en chemin on peut avoir de multiples constantes différentes possibles (cf. exemple 6.2.1).

**Exemple 6.2.1** *Considérons le cas des règles de paraphrase syntaxique profonde. Elles se présentent sous la forme d'une équivalence entre deux structures d'arbre. On a donc quelque chose de la forme* exp1 ≡ exp2. *Comme, dans les lexiques, on ne peut affecter à une expression une autre expression, j'ai eu recours à la transduction pour modéliser cette équivalence (cf. figure 11 page suivante qui illustre le cas particulier de la règle lexicale 24 (cf. exemple 6.4.1 page 24)). Pour toute règle lexicale rlex i, je crée*

*une constante rlex-i dans* $\Sigma_{lexeq}$*, deux constantes ri-1 et ri-2 dans* $\Sigma_{comp}$*, et deux expressions expi-1 et expi-2 dans* $\Lambda(\Sigma_{deep\text{-}rel})$ *(cf. figure 11) telles que :*

— $\mathcal{L}_{compRel}(ri\text{-}1) = expi\text{-}1$ *et* $\mathcal{L}_{compRel}(ri\text{-}2) = expi\text{-}2$

— $\mathcal{L}_{compEq}(ri\text{-}1) = rlex\text{-}i = \mathcal{L}_{compEq}(ri\text{-}2)$

*On obtient donc bien l'équivalence voulue entre les deux expressions :* $expi\text{-}1 \equiv expi\text{-}2$.

*Pour le cas particulier de la règle lexicale 24, lors de la transduction (en rouge sur la figure 11) de* $\Sigma_{lexeq}$ *à* $\Sigma_{deep\text{-}rel}$*, tous les antécédents possibles de* $rlex\text{-}24$ *(ici* exp1 *et* exp2*) seront renvoyés.*

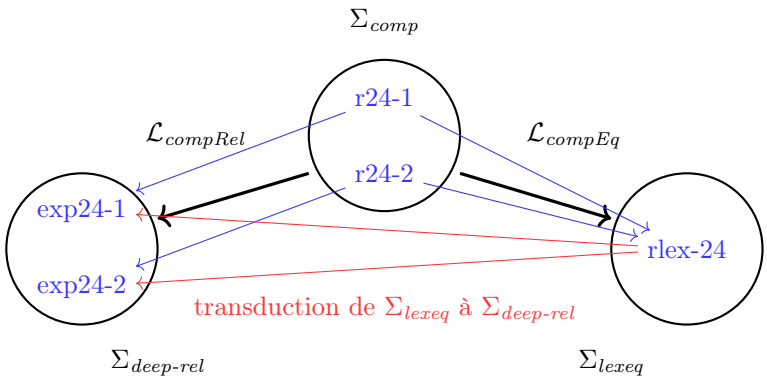



FIGURE 11 – Illustration de la transduction dans le cas de l'implémentation de la règle lexicale 24 de paraphrase syntaxique profonde

J'utilise cet avantage de la transduction pour la paraphrase syntaxique profonde, et pour la réalisation (et l'apparition) des FLs. Pour la paraphrase sémantique et la transition de la sémantique vers la syntaxe profonde (de $\Sigma_{semantic}$ à $\Sigma_{deep\text{-}rel}$), il s'agit d'une transduction "classique", tout comme pour la transition de la syntaxe profonde à la syntaxe de surface (de $\Sigma_{\textit{finfl}}$ à $\Sigma_{surface\text{-}tree}$).

**Signature abstraite "omnisciente" :** À plusieurs reprise ($\Sigma_{deep\text{-}syntactic}$ et $\Sigma_{surface\text{-}syntactic}$), la signature abstraite des ACGs utilisées ne précise pas les relations syntaxiques (profondes pour $\Sigma_{deep\text{-}syntactic}$, et de surface pour $\Sigma_{surface\text{-}syntactic}$) entre le prédicat et ses arguments. En effet, j'ai fait le choix de faire comme si on savait de quelle relation il s'agissait. Ces relations sont détaillées immédiatement après, respectivement par $\mathcal{L}_{dsyntRel}$ dans $\Sigma_{deep\text{-}rel}$, et $\mathcal{L}_{stree}$ dans $\Sigma_{surface\text{-}tree}$.

Cela m'a permis d'utiliser la transduction autour de ces signatures d'une part, et, d'autre part, la transition que je voulais effectuer avait besoin d'une étape intermédiaire qui représentait les informations de manière simplifiée sans toutefois perdre des informations. En effet, la structure de données à modéliser changeait (pour $\Sigma_{deep\text{-}syntactic}$, on avait un graphe sémantique dans $\Sigma_{semantic}$ et on veut un arbre de dépendance en syntaxe profonde dans $\Sigma_{deep\text{-}rel}$, et pour $\Sigma_{surface\text{-}syntactic}$, on avait un arbre de dépendance de syntaxe profonde dans $\Sigma_{\textit{finfl}}$ et on voulait un arbre de dépendance de syntaxe de surface dans $\Sigma_{surface\text{-}tree}$), il fallait donc utiliser les informations de la structure précédente pour en créer une nouvelle. Avoir cette représentation simplifiait permet donc une transition plus facile et plus libre.

On peut d'ailleurs remarquer que les deux transductions que j'évoque ici ont été caractérisées de "classiques" au paragraphe précédent. En effet, il n'y a pas ici plusieurs manières de passer d'une signature objet à une autre : pour un élément de $\Sigma_{deep\text{-}rel}$, son antécédent (via $\Sigma_{deep\text{-}syntactic}$) dans $\Sigma_{semantic}$ est unique. Le même phénomène se retrouve pour $\Sigma_{surface\text{-}syntactic}$.

**Structure communicative :** Pour des soucis de simplification de la modélisation, je n'ai pas représenté la structure communicative de chacun des niveaux de représentation traités, mais uniquement la structure principale. En effet, la structure communicative aurait demandé d'annoter les structures représentées par ma modélisation, ce qui aurait compliqué fortement la tâche, et n'était pas le but premier de ce stage. Ce dernier visait plutôt à étudier ce qui était possible de faire, et non comment modéliser la TST avec les ACGs de manière parfaite. C'est d'ailleurs l'une des limitations de ce projet (cf. partie 8.2), je n'ai pas pris en compte les structures autres que les structures principales, y compris la structure communicative, qui joue pourtant un rôle essentiel.

En effet, c'est elle qui détermine, entre autres, le thème et le rhème dans le graphe sémantique, le nœud dominant du graphe sémantique, et le nœud qui sera racine de l'arbre de syntaxe profonde. J'ai donc choisi ces nœuds dominants arbitrairement, selon ce qui me paraissait le plus approprié ; les phrases (ou exemples) générables par cette modélisation étant peu nombreuses et plutôt courtes, cela n'a pas posé de problème, mais en posera un si cette modélisation venait à être étendue avec un lexique (dans le sens d'un ensemble de mots utilisés et connus, et non le lexique d'une ACG) plus fourni.

**TAG :** La TST utilisant des structures d'arbres, et les ACGs étant très adaptées pour la modélisation des TAG (Pogodalla 2017a), je me suis inspirée à de nombreuses reprises de principes de modélisation utilisés pour les TAG, en particulier en ce qui concerne les groupes adverbiaux (cf. les parties suivantes, où cela est plus détaillé).

**Limitations des ACGs :** Les ACGs présentent quelques limitations, qui se répercutent dans ce projet. En effet, toutes les égalités dans les ACGs sont modulo $\alpha$, $\beta$ ou $\eta$-reduction, on ne peut donc pas identifier toutes les égalités, y compris les équivalences logiques, telles la réflexivité du "et" logique. Aussi, si, à un endroit, j'ai écrit quelque chose de la forme `A & B`, et à un autre `B & A`, avec `A` et `B` identiques, rien ne permettra de savoir dans les ACGs que ces expressions sont équivalentes. Ce phénomène est apparu en particulier lors de la composition de graphes sémantiques élémentaires. En effet, il ne trouvait parfois pas tous les antécédents possibles, car, en composant les formules, deux termes se retrouvaient intervertis.

Ce phénomène revient dans les modules de la TST qui comportent des règles d'équivalence entre termes. Cependant, ces équivalences étant plus simples que les équivalences logiques, j'ai pu le contourner en composant des lexiques grâce à la transduction.

**Remarque 6.2.2** *Dans les types non atomiques que j'ai utilisés, les symboles $\multimap$ et $\to$ apparaissent. La seule différence entre ces deux symboles est que $\multimap$ est linéaire (cf. définition 4.1.3 page 6) alors que $\to$ ne l'est pas forcément. Aussi, $n \multimap t$ est par exemple un type linéaire, mais $n \to t$ pas forcément. Dans ce deuxième cas, le terme abstrait (de type $n$ donc), pourra être utilisé plusieurs fois dans le $\lambda$-terme.*

## 6.3 Modélisation de la sémantique

La partie de ma modélisation qui manipule la sémantique est constituée de la représentation sémantique, soit de la signature $\Sigma_{semantic}$ (zone rouge de la figure 10 page 15), et d'une partie du module sémantique. Cette première partie du module sémantique est constitué de la signature $\Sigma_{deep\text{-}syntactic}$ du lexique $\mathcal{L}_{sem}$ quittant $\Sigma_{deep\text{-}syntactic}$ et pointant sur $\Sigma_{semantic}$.

Je détaille ici les objectifs de représentation, puis mon implémentation de cette partie sémantique, et enfin les problèmes auxquels j'ai été confrontée ainsi que les choix que j'ai fait.

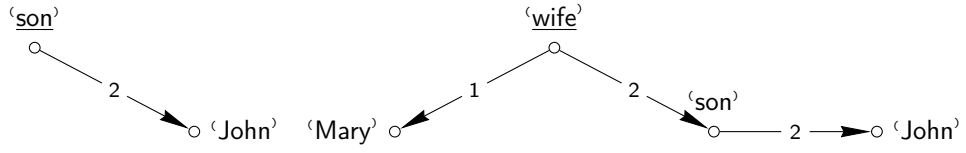
### 6.3.1 Objectifs

**Représentation :** Le but de cette partie sémantique est d'une part de modéliser la représentation sémantique **SemR**, mais aussi d'effectuer la paraphrase sémantique et de préparer la transition de la représentation sémantique **SemR** à la représentation de syntaxe profonde **DSyntR**.

Au niveau de la représentation sémantique, comme détaillé en section 5.2.1, il s'agit de représenter des graphes, dont les nœuds ont des étiquettes, et les arêtes des labels. La structure de données utilisée doit donc permettre cela. En outre, ACGtk reposant sur du $\lambda$-calcul typé, les types utilisés doivent permettre à un graphe d'avoir des sous-graphes, et à un graphe de devenir un sous-graphe.

**Exemple 6.3.1** *La phrase nominale* « John's son » *doit donner lieu à un graphe, mais le graphe représentant la phrase* « Mary, the wife of John's son »*, comporte un sous-graphe qui est exactement le graphe précédent (cf. figure 12b page suivante).*

**Arguments optionnellement exprimables :** En outre, les emplacements des arguments d'un prédicat sémantique peuvent être optionnellement exprimables (Mel'čuk et al. 2015), bien qu'obligatoires. En effet, on peut dire « *Mary, the wife of John* » ou « *John's wife* ». Ces deux phrases sont correctes, et il s'agit du même prédicat « *wife* » (ils ont tout deux exactement le même sens), celui qui admet comme



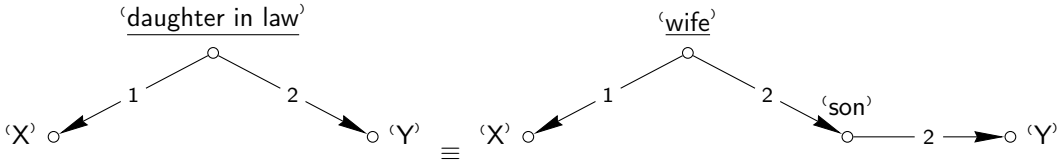

(a) « *John's son* » (b) « *Mary, the wife of John's son* »

FIGURE 12 – Illustration de la compositionnalité des graphes sémantiques

premier argument sémantique (« *Mary* » dans la première phrase) un argument optionnellement exprimable. Il faut donc permettre à un emplacement d'exister, mais d'être possiblement vide. De la même manière, certains arguments sont obligatoires : on doit pouvoir imposer cette obligation d'expression (en effet, on ne peut pas dire « *Mary is the wife.* », il manque « *of X* » où X est une personne).

**Paraphrase sémantique :** En ce qui concerne la paraphrase sémantique, on peut distinguer deux catégories. Quoi qu'il arrive, la paraphrase sémantique consiste en une définition plus précise d'un sémantème ou de plusieurs sémantèmes dans la phrase (cf. exemples 6.3.2 et 6.3.3). Cependant, la phrase de définition d'une UL proposée par le DEC (cf. section 5.4 et exemple 5.4.2 page 13) peut être naturelle pour un locuteur natif ou non. Dans ce dernier cas (c'est le cas de la définition donnée en exemple 5.4.2 page 13), le graphe plus précis doit exister, mais ne doit pas pouvoir générer une phrase traduite littéralement par la suite, mais uniquement la phrase utilisant l'entrée lexicale qu'il définit (Mel'čuk et al. 2013). Ces paraphrases définitionnelles sont données dans le DEC (cf. section 5.4).

**Exemple 6.3.2 (Mel'čuk et al. 2013)** *Pour* « X the daughter in law of Y »*, on obtient la paraphrase* « X the wife of Y's son »*. Ces deux phrases sont licites et naturelles : les deux graphes doivent donc être liés d'une part au sémantème* « daughter in law »*, mais le second doit aussi pouvoir générer la phrase* « X the wife of Y's son » *(cf. figure 13b).*



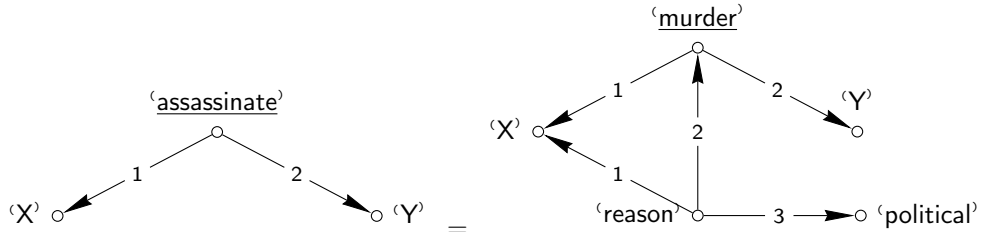

(a) « *X the daughter in law of Y* » (b) « *X the wife of Y's son* »

FIGURE 13 – Illustration de l'équivalence de deux graphes pour la paraphrase sémantique

**Exemple 6.3.3 (Mel'čuk et al. 2013)** *Pour l'UL* ASSASSINATE *, utilisée ainsi*[2] *:* « X assassinate Y »*, on obtient la paraphrase* « X murder Y for a political reason. »*. Cette deuxième phrase est correcte, mais on lui préfèrera* « assassinate »*. Il faudrait donc que les deux graphes génèrent la phrase avec* « assassinate »*, et que le second graphe puisse aussi générer la seconde phrase, mais que lorsqu'on a le choix, on préfère* « X assassinate Y. » *à* « X murder Y for a political reason. »*.*

**Transition vers la représentation de syntaxe profonde :** Finalement, après l'étape de paraphrase sémantique, vient l'étape de transition de la représentation sémantique à la représentation de syntaxe profonde (cf. section 6.4.1). Cette etape doit permettre de lier chaque sémantème à son lexème associé (opération de lexicalisation), et elle doit construire un arbre de dépendance syntaxique à partir de la représentation sémantique. Aussi, tous les graphes sémantiques valables (donc ayant une structure correcte, indépendamment du sens exprimé), devraient pouvoir donner lieu à une représentation de syntaxe profonde, à partir du moment où l'arbre de dépendance correspondant est lui aussi correct du point de vue de sa structure. L'objectif est ici de préparer cette transition, que j'expliciterai en section 6.4.2.

2. D'après l'exemple récurrent disponible dans les deux volumes de I.A. Mel'čuk, 2012 et 2013.




(a) « *X assasinate Y.* » (b) « *X murder Y for a political reason.* »

FIGURE 14 – Illustration de l'équivalence de deux graphes pour la paraphrase sémantique lorsqu'un sens de paraphrase est préféré

### 6.3.2 Implémentation

Afin de modéliser cela, j'ai implémenté une signature propre au niveau de représentation sémantique **SemR** de la TST, qui est $\Sigma_{semantic}$ (cf. figure 37 page 56).

- Deux types y sont définis : les types $n$ (pour "nœud") et $t$ (pour "truth value").
- J'y définis également la relation du "et" logique $\wedge$.
- Dans cette signature, j'ai choisi de représenter tous les prédicats sémantiques de la même manière : ils sont tous de type $n \to t$, soit associent à un nœud une valeur de vérité. Ainsi, chaque prédicat sémantique représente un nœud, donc un graphe élémentaire.
- J'ai également défini des relations $R1$ (pour le premier actant sémantique), $R2$ (pour le second actant sémantique), etc. de type $n \multimap n \multimap t$, qui relient deux nœuds pour donner une valeur de vérité.

**Exemple 6.3.4** *Le graphe sémantique de la figure 12b page précédente sera représenté par le $\lambda$-terme suivant : $\lambda\ e0.\ \exists\ ex\ ey.\ (wife\ e0) \wedge (R1\ e0\ ex) \wedge (R2\ e0\ ey) \wedge (mary\ ex) \wedge (\exists ex'\ ey'.\ (son\ ey) \wedge (R1\ ey\ ex') \wedge (R2\ ey\ ey') \wedge (\exists\ v.\ v\ ex') \wedge (john\ ey'))$ de type $n \to t$. L'obtention de ce $\lambda$-terme est détaillé dans le paragraphe suivant.*

Ainsi, un graphe sémantique sera représenté de la manière suivante. Prenons comme exemple la phrase de l'exemple 6.3.4 :

1. Le nœud sémantiquement dominant est abstrait. Ici c'est ‘wife’, on a donc $\lambda\ e0.\ wife\ e0$.
2. Si ce nœud dominant prend des arguments sémantiques, il faut leur attribuer un nœud, et les rattacher au nœud dominant. Ici, ‘wife’ prend deux arguments sémantiques ; ‘Mary’ en premier argument et ‘son’ en second argument : le $\lambda$-terme précédent devient $\lambda\ e0.\ \exists\ ex\ ey.\ (wife\ e0) \wedge (R1\ e0\ ex) \wedge (R2\ e0\ ey) \wedge (mary\ ex) \wedge (son\ ey)$.
3. Si ces nouveaux sémantèmes ont des arguments sémantiques, on répète l'étape 2 en remplaçant "nœud dominant" par le sémantème en question. Ici ‘son’ a un argument sémantique (‘John’), on obtient donc finalement le $\lambda$-terme de l'exemple 6.3.4.

En liant les graphes par les relations décrites ci-dessus, on obtient donc des graphes non élémentaires, eux aussi de type $n \to t$. Ce choix permet de composer des graphes très facilement (cf. exemple 6.3.5), indépendamment de contraintes telles que "est-ce que cela a un sens", "quelle phrase sera générée par la suite", etc. Le but de tout cela est qu'un graphe soit vu comme des structures relationnelles. Ils sont ainsi représentés par des formules logiques, qui ont l'avantage d'être évaluables, et comparables.

**Exemple 6.3.5** *Le terme $c_wife\ (EXP\ c_mary)\ (c_son\ IMP\ c_john)$ [3] du vocabulaire $\Sigma_{deep\text{-}syntactic}$ permet d'obtenir le graphe sémantique illustré en figure 12b page précédente. En effet :*
*$\mathcal{L}_{sem}(c_wife\ (EXP\ c_mary)\ (c_son\ IMP\ c_john) = \lambda\ e0.\ \exists\ ex\ ey.\ (wife\ e0) \wedge (R1\ e0\ ex) \wedge (R2\ e0\ ey) \wedge (mary\ ex) \wedge (\exists\ ex'\ ey'.\ (son\ ey) \wedge (R1\ ey\ ex') \wedge (R2\ ey\ ey') \wedge (\exists\ v.\ v\ ex') \wedge (john\ ey'))$. Cette égalité s'obtient par $\beta$-réduction à partir de l'implémentation des sémantèmes dans $\mathcal{L}_{sem}$ (cf. figure 37 page 56).*

3. Les constantes $IMP$ et $EXP$ seront introduites par la suite. Elles différencient les arguments sémantiques obligatoirement exprimables de ceux optionnellement exprimables. On peut dans un premier temps faire abstraction de ces constantes.

Le lien entre un prédicat et ses arguments se fait au niveau du lexique $\mathcal{L}_{sem}$, qui a pour vocabulaire abstrait $\Sigma_{deep\text{-}syntactic}$ et pour vocabulaire objet $\Sigma_{semantic}$. En effet, $\Sigma_{semantic}$ pourrait s'apparenter à un sac de briques de Lego, où les briques sont les sémantèmes et relations sémantiques, $\Lambda(\Sigma_{deep\text{-}syntacic})$ pourrait s'apparenter à l'ensemble des constructions finies, et $\mathcal{L}_{sem}$ à la personne qui construit. Autrement dit, dans $\Sigma_{semantic}$ chaque sémantème possède des emplacements disponibles pour qu'un autre sémantème s'y "branche" selon la définition sémantique de ce premier sémantème selon le DEC, mais aucun branchement n'a lieu dans la signature. C'est le lexique $\mathcal{L}_{sem}$ qui fait ce lien. C'est également lui qui va contraindre quelles représentations sémantiques pourront donner suite à une représentation de syntaxe profonde ou non. En effet, ces représentations sémantiques considérées comme "correctes" [4] seront en lien avec un élément de $\Lambda(\Sigma_{deep\text{-}syntactic})$, contrairement à celles "incorrectes", qui n'auront aucun lien avec cet ensemble, et ne pourront donc rien générer.

La signature $\Sigma_{deep\text{-}syntactic}$ est un "condensé" de la syntaxe profonde (cf. figure 39 page 57). On part du principe que l'on sait quelle relation existe entre le prédicat et chacun de ses arguments, et on donne au prédicat ses arguments dans le bon ordre vis-à-vis de ces relations implicites. Aussi, chaque prédicat (ou entrée lexicale) y est défini par un type atomique ($G$) s'il ne prend pas d'arguments (et n'a donc pas de sous-graphe dans la représentation sémantique, ou de sous-arbre dans la représentation syntaxique profonde), et par un type non atomique (tel que $G \multimap g \multimap G$ dans le cas de deux arguments) s'il prend plusieurs arguments (cf. figure 39 page 57).

La transition vers la syntaxe profonde se fait grâce au lexique $\mathcal{L}_{dsyntRel}$, qui prend les expressions "condensées" de $\Lambda(\Sigma_{deep\text{-}syntactic})$ et construit les arbres détaillés de syntaxe profonde dans $\Lambda(\Sigma_{deep\text{-}rel})$ (cf. section 6.4.2), qui est la signature modélisant de niveau de représentation de syntaxe profonde **DSyntR** (cf. section 6.4.2).

### 6.3.3 Problèmes rencontrés

**Paraphrase sémantique :** La paraphrase sémantique (à comprendre comme une équivalence sémantique, cf. section 5.3.1) est basée sur les définitions des prédicats sémantiques, qui sont donc équivalentes (en terme de sens) au prédicat sémantique en question. Un problème se pose alors : chaque prédicat sémantique possède une définition, constituée de prédicats sémantiques, qui possèdent eux-même une définition, constituée de prédicats sémantiques, etc. La TST a été pensée pour que chaque sens soit défini uniquement par des sens "plus simples" (Mel'čuk et al. 2012) (cf. sections 5.2.1 et 5.4). Les boucles dans la définition sont donc exclues, mais la suite de paraphrases possibles est très grande jusqu'aux sens élémentaires, sans compter qu'une définition est généralement composée de plusieurs mots. On peut donc "déplier" la définition d'un mot, ou de plusieurs, ce qui laisse encore plus de possibilités.

J'ai donc décidé de ne mettre qu'une étape de paraphrase sémantique, qui fait le lien entre une UL et son sens, et/ou si il est complexe, de lier cette UL également à sa définition détaillée, en détaillant la définition le plus possible. Les étapes de "dépliage" intermédiaires ne sont donc pas représentées, et inexistantes ici. J'ai testé cela pour la phrase *X assassinate Y*, qui peut avoir les deux graphes sémantiques vu en exemple 6.3.3 page 18.

Pour chaque entrée lexicale, la règle d'encodage est donc la suivante : si il existe une définition irréductible autre que l'entrée en elle même (si l'entrée n'est pas un sens élémentaire), sa représentation sémantique sera la représentation de sa définition irréductible, selon les règles d'encodage d'une structure sémantique (cf. section 6.3.2). Sinon (l'entrée lexicale est un sens élémentaire), la représentation de l'entrée sémantique est cette entrée elle-même. Cependant, ce stage étant d'une durée limitée, pour des soucis de simplification, parce que je n'avais pas toutes les ressources voulues, et parce que la TST est complexe avec beaucoup de phénomènes à étudier, je n'ai appliqué cette règle d'encodage que pour une entrée lexicale (ASSASSINATE ), et ai considéré toutes les autres comme étant des sens élémentaires, bien que cela ne soit pas forcément le cas.

Aussi, un graphe sémantique étant lié à une unité lexical grâce au lexique $\mathcal{L}_{sem}$, je n'ai pas implémenté d'exemple de paraphrase où la définition n'est pas naturelle. En effet, ce cas revient à empêcher le graphe définitionnel de produire une phrase le traduisant littéralement, autrement dit à supprimer (ou ne pas faire figurer) une entrée du lexique. J'ai préféré me concentrer sur d'autres exemples et cas particuliers.

4. Ici "correcte" signifie "qui a un antécédent en syntaxe profonde" (dans le vocabulaire ($\Sigma_{deep\text{-}rel}$)) donc.

**Nœud communicativement dominant et conséquences :** Comme vu en section 5.2.1, un graphe sémantique a un nœud dominant. Ce nœud dominant est déterminé grâce à des règles spécifiques (Mel'čuk et al. 2012 ; Mel'čuk et al. 2013). Ce nœud dominant est celui abstrait dans mon implémentation. L'exemple 6.3.4 page 19 illustre la réponse d'ACGtk à la commande (elle aussi illustrée exemple 6.3.4 page 19) d'analyse par $\mathcal{L}_{sem}$ de la représentation dans $\Sigma_{deep\text{-}syntactic}$ de « *Mary the wife of John's son* ». On y voit comment le graphe de la figure 12b page 18 y est encodé, avec l'abstraction du nœud $e0$ qui est ici le nœud dominant.

Pour les graphes représentant une seule UL, donc en relation directe avec le prédicat qu'ils représentent, le nœud dominant est a priori le bon. En effet, je n'ai pas représenté la structure communicative pour des soucis de simplification de ma solution (cf. partie 6.2), or ce nœud est déterminé à partir de la structure communicative. Il s'en suit que lorsqu'on compose des graphes, le nœud dominant est déterminé arbitrairement (comme étant le nœud dominant du graphe père ou racine), et n'est donc pas forcément celui qu'il aurait été dans le cadre d'une modélisation avec une structure communicative. Cela ne pose cependant pas de problème ici, car les phrases modélisées restent assez simples, mais pourrait en devenir un si on étend ce modèle sans y ajouter une structure communicative.

Ce choix d'implémentation de n'abstraire que le nœud dominant implique un autre problème de modélisation. En effet, comme il est le seul abstrait, il est le seul accessible. Ainsi, si on veut composer deux graphes (ce qui est a priori autorisé sans contraintes particulières tant que les structures des graphes sont respectées, et que le branchement a lieu à l'emplacement d'une variable du graphe parent), on ne peut que "brancher" le nœud dominant d'un graphe au graphe parent, et non au autre nœud de ce graphe.

**Exemple 6.3.6** *Aussi, le graphe illustré en figure 15 n'est pas réalisable, ou plutôt ne donnera pas de suite lors de la génération.*

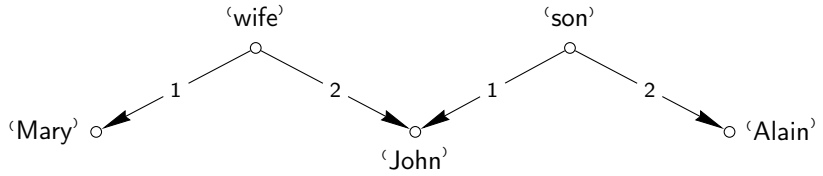
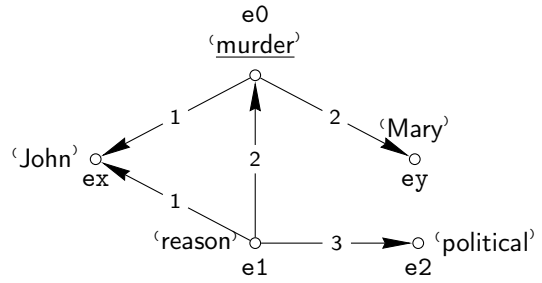



FIGURE 15 – « *Graphe sémantique irréalisable* »

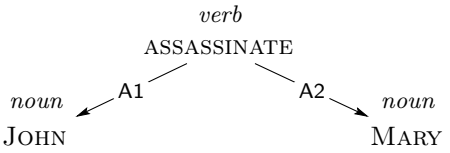
**Architecture de la modélisation :** Afin de préparer la transition (et de pouvoir la faire par la suite) entre les graphes sémantiques ($\Sigma_{semantic}$) et les arbres de syntaxe profonde ($\Sigma_{deep\text{-}rel}$), j'ai ajouté une signature intermédiaire : $\Sigma_{deep\text{-}syntactic}$. Cette signature est assez équivalente à $\Sigma_{deep\text{-}rel}$ (cf. section 6.4.2), sauf qu'elle ne précise pas les relations syntaxiques profondes (cf. exemple 6.3.7). $\Sigma_{deep\text{-}syntactic}$ est la signature liée au langage abstrait de la grammaire comprenant $\Sigma_{deep\text{-}syntactic}$, $\Sigma_{deep\text{-}rel}$, et $\mathcal{L}_{dsyntRel}$, on considère savoir dans $\Sigma_{deep\text{-}syntactic}$ les liens entre les différentes constantes. Aussi, dans le terme $c_assassinate\ c_john\ c_mary$ du vocabulaire $\Sigma_{deep\text{-}syntactic}$, on sait que $c_john$ est l'actant 1 de $c_assassinate$ et que $c_mary$ en est le second actant. C'est $\mathcal{L}_{dsyntRel}$ qui rendra cela explicite en le représentant dans $\Lambda(\Sigma_{deep\text{-}rel})$ par un $\lambda$-terme explicitant les bonnes relations syntaxiques profondes.

**Exemple 6.3.7** *Dans le vocabulaire ($\Sigma_{deep\text{-}syntactic}$), la phrase* « John assassinate Mary. » *sera représentée comme cela : $c_assassinate\ (EXP\ c_john)\ c_mary\ :\ G$. En réalisant cette représentation grâce à $\mathcal{L}_{sem}$ dans $\Sigma_{semantic}$, le graphe figure 16a page suivante est obtenu. En l'analysant grâce à $\mathcal{L}_{dsyntRel}$ dans $\Sigma_{deep\text{-}rel}$, le graphe 16b page suivante est obtenu.*

**Arguments optionnellement exprimables :** Ensuite s'est posée la question de la modélisation des arguments optionnellement exprimables. Pour cela, je me suis inspirée de (Blom et al. 2011), et ai défini dans la signature $\Sigma_{deep\text{-}syntactic}$ deux types $G$ et $G'$, G pour les arguments obligatoirement exprimables et optionnellement exprimables respectivement, ainsi que deux constantes $IMP$ et $EXP$. $IMP$ laisse l'argument vide, et se manifeste dans le graphe sémantique par l'introduction d'un nœud d'étiquette $v$ quantifié existentiellement (cf. exemple 6.3.5 page 19). Par exemple, « *John's wife* » est une expression



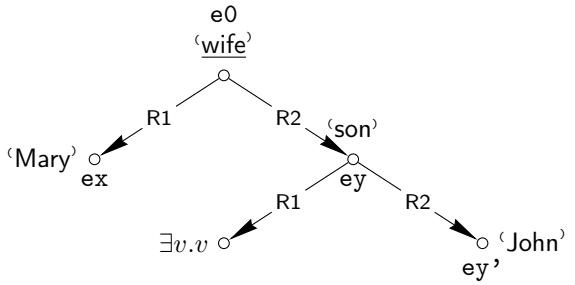

(a) Illustration de la représentation de la pharse « *John assasinate Mary.* » dans $\Sigma_{semantic}$, dont le $\lambda$-terme correspondant est : $\lambda\ e0.\ (\exists\ ex\ ey\ ez.\ (murder\ e0)\ \wedge\ (R1\ e0\ ex)\ \wedge\ (R2\ e0\ ey)\ \wedge\ (john\ ex)\ \wedge\ (mary\ ey)\ \wedge\ (\exists\ e1\ e2.\ (reason\ e1)\ \wedge\ (R1\ e1\ ex)\ \wedge\ (R2\ e1\ e0)\ \wedge\ (R3\ e1\ e2)\ \wedge\ (politics\ e2)))$



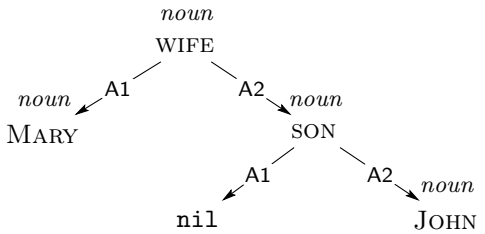

(b) Illustration de la représentation de la phrase « *John assasinate Mary.* » dans $\Sigma_{deep\text{-}rel}$, dont le $\lambda$-terme correspondant est : $verb\ ASSASSINATE\ ((A1\ *\ (noun\ JOHN\ void))\ \wedge\ (A2\ *\ (noun\ MARY\ void)))$

FIGURE 16 – Illustration des représentations dans $\Sigma_{semantic}$ et $\Sigma_{deep\text{-}rel}$ de « *John assassinate Mary.* »

dans laquelle le premier argument sémantique n'a pas été exprimé, alors qu'il l'est dans l'expression « *Mary, the wife of John* ».




(a) Représentation du graphe sémantique de « *Mary, the wife of John's son* » dans $\Sigma_{semantic}$, dont le $\lambda$-terme correspondant est : $\lambda\ e0.\ \exists ex\ ey.\ (wife\ e0)\ \wedge\ (R1\ e0\ ex)\ \wedge\ (R2\ e0\ ey)\ \wedge\ (mary\ ex)\ \wedge\ (\exists\ ex'\ ey'.\ (son\ ey)\ \wedge\ (R1\ ey\ ex')\ \wedge\ (R2\ ey\ ey')\ \wedge\ (\exists\ v.\ v\ ex')\ \wedge\ (john\ ey'))$




(b) Représentation de l'abre de syntaxe profonde de « *Mary, the wife of John's son* » dans $\Sigma_{deep\text{-}rel}$, dont le $\lambda$-terme correspondant est : $noun\ WIFE\ ((A1\ *\ (noun\ MARY\ void))\ \wedge\ (A2\ *\ (noun\ SON\ ((A1\ *\ nil)\ \wedge\ (A2\ *\ (noun\ JOHN\ void))))))$

FIGURE 17 – Illustration des représentations dans $\Sigma_{semantic}$ et $\Sigma_{deep\text{-}rel}$ de « *Mary the wife of John's son* »

**Groupes adverbiaux** J'ai travaillé sur un prototype de la modification adverbiale, à savoir le groupe adverbial « *for political reason* », qui est donc un cas spécifique parmi les groupes adverbiaux. Cependant, ce qui suit se veut généralisable pour tous les groupes adverbiaux. L'inclusion de ce groupe adverbial dans la modélisation ci-dessus a posé quelques soucis. En effet, on a vu (cf. exemple 6.3.3 page 18) lors de la paraphrase de « *X assassinate Y* », que l'une des paraphrase se traduisait par « *X murder Y for a political reason* ». Seulement, le graphe sémantique n'est pas un arbre, et ‘for a political reason’ n'est pas un argument sémantique à proprement parler de ‘murder’. En effet, ‘reason’ prendra ‘X’, ‘murder’ et ‘political’ comme arguments, tandis que ‘murder’ prendra ‘X’ et ‘Y’ comme arguments. Les deux prédicats et leurs arguments sont entremêlés ; traiter le groupe adverbial comme un simple troisième argument n'était donc pas possible.

Pour cela, je me suis inspirée de (Pogodalla 2017a), et ai repris un principe utilisé dans les TAG : les adjectifs y sont représentés par un modificateur de type atomique interprété par une fonction.

J'ai donc définit $c_for_pol_reas$ de manière à l'utiliser similairement. Les figures 37 page 56 et 41 page 59 montrent comment ce groupe adverbial « *for a political reason* » est implémenté dans les signatures respectives $\Sigma_{semantic}$ et $\Sigma_{deep\text{-}rel}$ par les lexiques respectifs $\mathcal{L}_{sem}$ et $\mathcal{L}_{dsyntRel}$.

**Exemple 6.3.8** *Ainsi, la phrase* « John murder Mary for a political reason. » *sera représentée de la manière suivante dans* $\Sigma_{deep\text{-}syntactic}$ *:* $c_for_pol_reas\ (\lambda\ A\ x.\ c_murder\ A\ x\ c_mary))\ (EXP\ c_john)\ :\ G$;. *Les représentations des implémentations figures 37 page 56 et 41 page 59 adaptées à cette phrase sont illustrées en figure 16a page précédente et figure 18.*

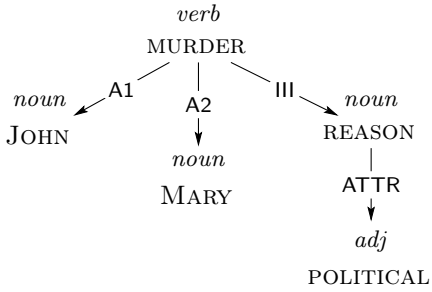



FIGURE 18 – Illustration de la représentation (de syntaxe profonde) de la phrase « *John murder Mary for a molitical reason.* » dans $\Sigma_{deep\text{-}rel}$ , dont le $\lambda$-terme correspondant est $verb\ MURDER\ (((A1\ *\ (noun\ JOHN\ void))\ \wedge\ (A2\ *\ (noun\ MARY\ void)))\ \wedge\ (A3\ *\ (noun\ REASON\ (ATTR\ *\ (adj\ POLITICAL\ void)))))$

Bien entendu, il a fallu prendre en compte les types des différents arguments, s'ils étaient obligatoirement exprimables ou non. Ainsi, le type de $c_for_pol_reas$ dans la signature $\Sigma_{deep\text{-}syntactic}$ est $(MOD \multimap G' \multimap G) \multimap G' \multimap G$ (cf. figure 39 page 57), car il prend en argument le verbe auquel il est rattaché et les arguments communs à ce verbe et à lui (ici il n'y en a qu'un, le sujet $X$). Ce verbe aura déjà tous ses emplacements complétés, excepté l'argument commun (ici $c_john$ dans l'exemple 6.3.8) et l'emplacement de type $MOD$ pour les modificateurs adverbiaux.

Cet argument laissé libre de type $MOD$ servira au niveau de la syntaxe profonde. La bonne compréhension la suite de ce paragraphe nécessite la lecture des parties 6.4.1 et 6.4.2 qui suivent ; elle se trouve donc en section 6.4.2.

## 6.4 Modélisation de la syntaxe profonde

Je considère cette modélisation constituée des éléments suivants :

— la signature $\Sigma_{deep\text{-}rel}$,

— la deuxième partie du module sémantique (à savoir la transition du niveau de représentation sémantique au niveau de représentation syntaxique profond) constituée de $\Sigma_{deep\text{-}rel}$ et du lexique $\mathcal{L}_{dsyntRel}$ pointant sur la signature $\Sigma_{deep\text{-}rel}$,

— la troisième partie du module sémantique, à savoir la zone concernant la paraphrase syntaxique profonde (en vert sur la figure 10 page 15),

— et la zone concernant les FLs (en violet sur la figure 10 page 15).

Cette troisième partie du module sémantique est constituée des deux signatures $\Sigma_{comp}$ et $\Sigma_{lexeq}$, et des deux lexiques $\mathcal{L}_{compRel}$ et $\mathcal{L}_{compEq}$.

### 6.4.1 Objectifs

**Représentation :** Le but de cette partie de syntaxe profonde est multiple. En effet, il s'agit ici de modéliser le niveau de représentation syntaxique profonde, ainsi qu'une part du module sémantique (la transition du niveau de représentation sémantique au niveau de représentation de syntaxe profonde, et la partie de paraphrase syntaxique), et de préparer la transition vers la syntaxe de surface.

Comme explicité en section 5.2.2, les structures de données utilisées en syntaxe profonde sont des arbres de dépendance, ayant pour racine le nœud communicativement dominant. Aussi, la modélisation utilisée ici doit permettre de représenter des arbres, dont les nœuds sont étiquetés par des lexèmes profonds, et donc les arrêtes sont étiquetées par les relations syntaxiques profondes. De plus, similairement à la modélisation sémantique, les arbres de syntaxe profonde contiennent des sous-arbres, qui peuvent eux-même être un arbre dans un autre contexte ou une autre phrase. La modélisation choisie doit donc permettre cette composition.

**Transition de la repésentation sémantique à la représentation de syntaxe profonde :** Dans la modélisation sémantique, la structure de données privilégiée est un graphe. Ici, c'est un arbre de dépendance. Il faut donc que cette transition permette le changement de structure de données. Comme le principe des ACGs comme de la TST n'est pas de modifier une structure, mais d'en créer une nouvelle à partir de la structure existante, c'est à priori facile. Cependant, comme on l'a vu en section 6.3.1, tous les graphes sémantiques ne génèreront pas un arbre de syntaxe profonde, et cette régulation des graphes a lieu grâce à la signature $\Sigma_{deep\text{-}syntactic}$. Aussi, il est nécessaire d'avoir un lexique permettant de passer des éléments de $\Sigma_{deep\text{-}syntactic}$ à une représentation de syntaxe profonde.

**Paraphrase syntaxique profonde :** La paraphrase syntaxique profonde est régulée par deux types de règles principalement (cf. section 5.3.1) : les règles lexicales, soutenues par les règles de restructuration. L'exemple 6.4.1 donne un exemple pour chacune de ces deux règles. Cependant, il se peut qu'aucune règle de restructuration ne soit nécessaire ; il faudrait donc, lors de l'implémentation de ces règles de paraphrase, pouvoir remplacer un lexème par un autre sans modifier la structure lorsque seule une règle lexicale s'applique, et pouvoir modifier l'arbre en question, puis ajuster les lexèmes utilisés, lorsqu'une règle de restructuration s'applique. La figure 33 page 51 montre comment ces règles peuvent être appliquées.

**Exemple 6.4.1 (Mel'čuk et al. 2013)** *Voici la règle lexicale 24, qui utilise comme règle support la règle de restructuration 4. La figure 33b page 51 illustre comment cette règle s'applique.*

| *Règle lexicale **24** - utilise la règle support 4* | | | |
|---|---|---|---|
| $V_0$(L ) | ≡ | *Anti*($V_0$(L )) | + *Non* |
| $\alpha$ | | $\beta$ | $\gamma$ |

| *Règle de restructuration **4** - **ATTR-Split** supporte les règles lexicales 5 et 23 à 26* |
|---|
| $\beta$ <br> \| <br> $\alpha$ ↔ *ATTR* <br> ↓ <br> $\gamma$ |

**Fonctions lexicales :** Les FLs apparaissent au niveau de syntaxe profonde. Il faut donc les modéliser ici également. En outre, elles peuvent apparaitre également lors de la paraphrase syntaxique profonde. Il faudra donc que cette modélisation tienne compte des FLs, et traite celles-ci de la même manière qu'un lexème, tout en se rappelant que ce sont des fonctions, et donc pas des lexèmes à proprement parler (cf. figure 33 page 51) : elles seront amenées à être réalisées lors de la transition vers la syntaxe de surface.

De plus, lors de l'étape de paraphrase syntaxique profonde, de nouveaux arbres sont générés, et chacun de ces arbres peut donner lieu à une nouvelle phrase (une paraphrase), si les mots contenus dans le lexique le permettent (cf. annexe E). Il faut donc garder chacun des arbres intermédiaires obtenus au cas où ils donnent lieu à une nouvelle paraphrase.

**Transition vers la représentation de syntaxe de surface :** En syntaxe de surface, les FLs n'existent plus. Afin de préparer la transition vers la modélisation de la syntaxe de surface, il faut donc réaliser les FLs, soit les remplacer par un lexème. Comme, pour une même UL, une FL peut avoir plusieurs valeurs possibles, cette réalisation donnera autant d'arbres de syntaxe profonde qu'il y a de valeurs possible à cette FL. En outre, les arbres ayant une structure erronée (tel que celui illustré figure 19, obtenu lors de la paraphrase syntaxique profonde (cf. annexe D)) ne devront pas atteindre la modélisation de syntaxe de surface. Il faut donc filtrer les arbres de syntaxe profonde lors de cette transition, afin que seuls des arbres correctement construits possèdent une représentation de syntaxe de surface.

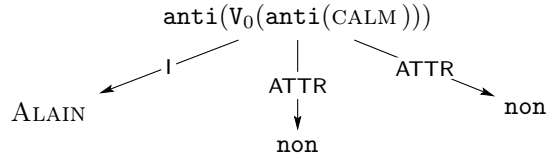



FIGURE 19 – arbre de syntaxe profonde ayant une structure erronée

### 6.4.2 Représentation syntaxique profonde

**Représentation :** Un arbre de syntaxe profonde possède des nœuds qui sont des lexèmes, et des branches vers ses nœuds fils, qui sont étiquetées par les relations syntaxiques profondes. Ils sont représentés dans la signature $\Sigma_{deep\text{-}rel}$ (cf. figure 40 page 58). L'obtention d'une telle représentation est détaillée après l'exemple 6.4.2.

Afin de modéliser cela, j'ai défini 7 nouveaux types. Le type *rel* correspond au type des relations syntaxiques profondes. Le type $T$ correspond à un arbre, et le type $Tb$ correspond à un arbre privé de sa racine, soit "un arbre à brancher". Ces arbres privés de leur racines sont constitués de la branche les rattachant à leur supposée racine, et du sous-arbre fils de cette supposée racine (cf. exemple 6.4.2). De plus, j'ai choisi de représenter la nature grammaticale des lexèmes à partir de la syntaxe profonde.Pour cela, j'ai donc introduit les types $V$ pour les verbes, $N$ pour les noms, $Adj$ pour les adjectifs et $Adv$ pour les adverbes.

J'ai introduit des constantes *verb*, *noun*, *adj* et *adv*, qui prennent un lexème (de type $V$, $N$, $Adj$, $Adv$ respectivement) et un arbre privé de sa racine (de type $Tb$) en argument, et revoient un arbre de type $T$.

J'ai également défini deux opérateurs, $\wedge$ et $*$, de types respectifs $Tb \to Tb \to Tb$ et $rel \to T \to Tb$. Le premier permet d'attacher deux arbres privés de leur racine en un seul arbre privé de sa racine, et le second permet de lier une relation syntaxique profonde à un arbre pour en faire un arbre privé de sa racine (cf. exemple 6.4.2).

La constante *void*, de type $Tb$ modélise l'absence de fils au nœud auquel elle est attachée : ce nœud est une feuille.

**Exemple 6.4.2** *La phrase* « John assassinate Mary the wife of Alain. » *sera représentée comme dans la figure 20 au niveau de la signature* $\Sigma_{deep\text{-}rel}$. *L'obtention de cette implémentation est détaillée dans le paragraphe suivant.*

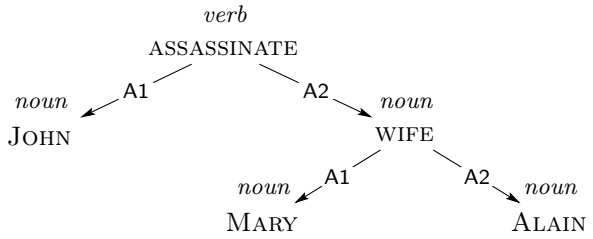



FIGURE 20 – Illustration de la représentation de syntaxe profonde de la phrase « *John assassinate Mary the wife of Alain.* » dans $\Sigma_{deep\text{-}rel}$

Ainsi, un arbre de syntaxe profonde théorique sera encodé de la manière suivante. Prenons comme exemple la phrase de l'exemple 6.4.2 :

1 La racine de de l'arbre est un lexème, qui doit donc être encodé dans $\Lambda(\Sigma_{deep\text{-}rel})$, avec comme type celui correspondant à la nature grammaticale du mot qui lui sera attribué en syntaxe de surface : $ASSASSINATE : V$; La modélisation commencera donc par la constante associée à la nature grammaticale. Dans le cas de l'exemple 6.4.2, on aura $verb\ ASSASSINATE$ ....

2 Si ce nœud racine a des fils, chacun de ces fils est un arbre, attaché à la racine par une branche étiquetée d'une relation syntaxique profonde. On attache donc ces arbres fils à la racine par les bonnes relations : $verb\ ASSASSINATE\ ((A1 * fils1) \wedge (A2 * fils2))$ .

3 Il reste à encoder les fils. Pour cela, répéter les étapes 1 et 2 autant de fois que nécessaire en remplaçant "racine" par "nœud père". On obtient finalement $verb\ ASSASSINATE\ ((A1 * (noun\ JOHN\ void)) \wedge (A2 * (noun\ WIFE\ (A1 * (noun\ MARY\ void) \wedge (A2 * (noun\ ALAIN\ void))))))$.

En outre, les fonctions lexicales respectent les natures grammaticales données dans les ouvrages (Mel'čuk et al. 2015) et (Mel'Čuk et Polguère 2021), que j'ai traduit par du typage. Par exemple, le synonyme d'un verbe sera un verbe et non un adjectif. Pour chaque FL que j'ai choisi d'implémenter, j'ai implémenté cette FL en prenant en compte ces natures grammaticales (cf. exemple 6.4.3 page suivante).

**Exemple 6.4.3** *Prenons le cas de la FL* `anti`*. J'ai implémenté anti_verb, qui est la restriction de* `anti` *aux verbes. Elle est de type* $V \multimap V$*. De même, j'ai implémenté anti_adj, de type* $Adj \multimap Adj$*.*

Ainsi, $anti_verb(assassinate)$ sera exactement de même type que $assassinate$, et pourra donc être traité de la même manière, conformément aux objectifs.

**Transition de la représentation sémantique à la représentation de syntaxe profonde :** Comme on a pu le voir précédemment, $\Sigma_{deep\text{-}syntactic}$ est une signature intermédiaire entre ces deux représentation, et qui prépare cette transition. C'est le lexique $\mathcal{L}_{dsyntRel}$ (cf. figure 41 page 59) qui va permettre cette dernière, en liant les constantes de $\Sigma_{deep\text{-}syntactic}$ à celles de $\Sigma_{deep\text{-}rel}$, en ajoutant les relations syntaxiques profondes entre les lexèmes, conformément à la description de la représentation d'un arbre de syntaxe profonde ci-dessus.

**Arguments optionnellement exprimables :** Les arguments sémantiques optionnellement exprimables, représentés à l'aide de $EXP$ et $IMP$ dans $\Sigma_{deep\text{-}syntactic}$ sont représentés de la manière suivante dans $\Sigma_{deep\text{-}rel}$ :

- $IMP$ : $\mathcal{L}_{dsyntRel}$ (cf. figure 41 page 59) va remplacer l'expression $\exists v.v$ (indiquant que l'argument sémantique n'est pas exprimé) par une constante $nil$. Ainsi, l'arbre de syntaxe profonde correspondant aura une branche ne contenant aucun lexème à son extrémité, bien qu'ayant une étiquette correspondant à une relation syntaxique profonde (cf. figure 17b page 22).
- $EXP$ : $\mathcal{L}_{dsyntRel}$ (cf. figure 41 page 59) va remplacer $EXP\ le_lexeme$ par l'expression correspondant à $le_lexeme$ directement dans $\Sigma_{deep\text{-}rel}$.

**Groupes adverbiaux** Ce paragraphe vient compléter celui se trouvant en fin de section 6.3.3. L'argument de type $MOD$ du groupe adverbial est spécifié dans $\Sigma_{deep\text{-}syntactic}$ dans la signature des verbe qu'il peut complémenter (ici uniquement $murder$, cf. figure 39 page 57, afin de pouvoir lui ajouter un modificateur quelconque). $c_for_pol_reas$ remplacera cet emplacement vaquant par une constante $I_MOD$ interprétée par $\mathcal{L}_{sem}$ dans $\Lambda(\Sigma_{semantic})$ par $true$ (cf. figure 37 page 56). En syntaxe profonde, cet emplacement vaquant est exploité afin de pouvoir adjoindre la branche contenant le groupe adverbial au verbe (cf. figure 41 page 59). En effet, les ACGs utilisant un typage fort, on ne peut pas rajouter une branche à un prédicat (c'est équivalent à déclarer un argument de plus dans la signature), sans l'avoir spécifié dans la signature au préalable. J'ai donc ajouté un type $MOD$ spécialement pour cela. Il est interprété en valeur de vérité dans $\Sigma_{semantic}$ et en arbre dans $\Sigma_{deep\text{-}rel}$, afin de pouvoir garder un emplacement vaquant qui ne pourrait cependant pas être utilisé à tort et à travers. (Cela permettrait par exemple d'ajouter un argument sémantique en plus, ce qui n'aurait aucun sens : le prédicat étant défini sémantiquement parlant en $\Sigma_{semantic}$, il n'admet pas d'autre argument sémantique. En ajouter un reviendrait à commettre une erreur, et n'aurait aucun sens. Cependant, en syntaxe profonde, le groupe adverbial se comporte comme un troisième argument du prédicat. Utiliser un autre type que le type usuel pour les graphes ou les arbres permet de contraindre les possibilités dans le lexique, tout en autorisant les bonnes constructions.)

Cette manière d'implémenter le groupe adverbial $c_for_pol_reas$ est généralisable et utilisable pour tous les modificateurs adverbiaux.

### 6.4.3 Paraphrase

**Implémentation :** Afin de traiter la paraphrase syntaxique profonde, j'ai écrit deux signatures $\Sigma_{comp}$ et $\Sigma_{lexeq}$, ainsi que deux lexiques $\mathcal{L}_{compRel}$ et $\mathcal{L}_{compEq}$ (cf. figure 10 page 15, zone verte). La structure de l'implémentation est similaire à la transition de la représentation sémantique ($\Sigma_{semantic}$) à la représentation de syntaxe profonde ($\Sigma_{deep\text{-}rel}$), car elle utilise également la transduction. Pour toute règle $i$ d'équivalence (ou de paraphrase) (cf. exemple 6.2.1 page 15), j'ai créé une constante $rlex\text{-}i$ dans $\Sigma_{lexeq}$, deux constantes $ri$-1 et $ri$-2 dans $\Sigma_{comp}$ et deux expressions $expi$-1 et $expi$-2 dans $\Sigma_{deep\text{-}rel}$ telles que :

- $\mathcal{L}_{compRel}(ri\text{-}1) = expi\text{-}1$ et $\mathcal{L}_{compRel}(ri\text{-}2) = expi\text{-}2$
- $\mathcal{L}_{compEq}(ri\text{-}1) = rlex\text{-}i = \mathcal{L}_{compEq}(ri\text{-}2)$

On obtient donc l'équivalence voulue : $\mathcal{L}_{compRel}(ri\text{-}1) = expi\text{-}1 \equiv expi\text{-}2 = \mathcal{L}_{compRel}(ri\text{-}2)$. Cela a le bon goût, lorsqu'on effectue la transduction de $\Sigma_{deep\text{-}rel}$ à $\Sigma_{lexeq}$, puis de $\Sigma_{lexeq}$ à $\Sigma_{deep\text{-}rel}$, de renvoyer les deux structures équivalentes $expi$-1 et $expi$-2, et non pas uniquement l'une des deux.

**Règles d'équivalence sur les fonctions lexicales :** Il existe des liens entre les fonctions lexicales (cf. figure 33c page 51). Les liens que j'ai traité sont tous des liens d'équivalence, que j'ai donc implémentés de la même manière que les règles de paraphrase syntaxique.

**Contrôle de la paraphrase :** Un problème s'est posé lorsque j'ai commencé à effectuer les commandes permettant la paraphrase syntaxique : quand s'arrêter ? En effet, dans la figure 33 page 51, on peut voir que l'application de la première règle fait apparaitre une structure, sur laquelle peut alors s'appliquer la deuxième règle. Seulement, avant d'appliquer la première règle, on ne pouvait pas appliquer la deuxième. Ainsi, en effectuant une boucle de paraphrase, soit en appliquant une règle, on peut faire apparaitre un arbre sur lequel une autre règle peut s'appliquer. Alors, on voudrait exécuter ces commandes tant que possible.

Cela revient malheureusement à une boucle infinie. Cependant, en étudiant les cas, je me suis rendue compte que des structures intuitivement [5] incorrectes apparaissaient (cf. figure 19 page 24) dans tous les cas (pour les quelques règles que j'ai traité au moins). On peut donc imaginer des règles définissant si oui ou non un arbre est correct, et un contrôleur, qui connaitrait ces règles, et fonctionnerait selon l'algorithme 1.

**Algorithme 1 :** Algorithme décrivant le contrôleur de la paraphrase syntaxique profonde

```
Données : A_0 ;                                          /* l'arbre initial */
A ← [A_0] ;                                  /* la liste des arbres de paraphrase */
n ← 1 ;                                                   /* longueur de A */
k ← 0;
continue = true ;
tant que continue faire
    j = 0;
    pour tout arbre A_j ∈ A faire
        appliquer à A_j les deux transductions Σ_deep-rel → Σ_lexeq et Σ_lexeq → Σ_deep-rel;
        pour chaque arbre A' obtenu faire
            si A' ∈ A alors
                ne rien faire;
            sinon si A' est incorrect alors
                ne rien faire;
            sinon
                A ← A + [A'];
            fin
        fin
    fin
    k ← longueur(A);
    si n == k alors
        continue ← false ;          /* aucun nouvel arbre correct n'a été obtenu */
    fin
fin
retourner A
```

Ce contrôleur n'est cependant pas le but premier de ce stage. Mes échantillons de tests étant petits, j'ai réussi à me passer de ce contrôleur, et ai préféré avancer sur le projet que m'atteler à son implémentation. Néanmoins, implémenter ce contrôleur est l'une des perspectives possibles (cf. section 8.2).

### 6.4.4 Fonctions lexicales

**Apparition des FLs :** Les fonctions lexicales apparaissent lors de la transition du niveau sémantique au niveau de syntaxe profonde, ou lors de la paraphrase syntaxique profonde. Dans cette modélisation, elles apparaissent de la même manière.

5. Je n'ai pas trouvé d'information claire à ce propos dans les trois volumes de Mel'čuk et al., je n'ai donc pas de certitudes concernant le caractère bien formé ou non de ces structures. J'ai choisi de les considérer mal formée, car c'est ce qui me paraissait le plus intuitif.

Le lexique $\mathcal{L}_{dsyntRel}$ est supposé pouvoir en faire apparaître, même si dans les exemples que j'ai traités, aucune FL n'apparait à ce niveau là. Cependant, ce lexique fonctionnant exactement comme voulu, tout porte à croire que s'il fallait en faire apparaître (c'est souvent le cas lorsqu'il y a besoin d'exprimer des verbes supports au niveau sémantique), cela ne poserait aucun problème.

Les règles de paraphrase syntaxique sont de plus implémentées d'une telle manière qu'elles font apparaitre les FLs dans les arbres lors de leur application.

**Réalisation des FLs :** Dans le module syntaxique profond, juste avant la transition vers le niveau de représentation de syntaxe de surface, les FLs sont réalisées, i.e. elles sont remplacées par l'une de leurs valeurs possibles. Pour cela, j'ai écrit la signature $\Sigma_{fl}$, qui, à un couple (FL, unité lexicale) associe l'une des unités lexicales associées à ce couple.

Cependant, un couple (FL, UL) peut avoir un ensemble d'unités lexicales possibles, et elles doivent être toutes disponibles. Par exemple, $\texttt{anti}(\text{CALM}) = \text{UPSET}$ et $\texttt{anti}(\text{CALM}) = \text{RESTLESS}$. En effet, l'utilisateur sera amené à choisir quelle phrase il préfère, mais ce n'est en aucun cas le rôle d'ACGtk. De plus, une même UL peut être associée à plusieurs couples (FL, UL). Par exemple, $\text{ASLEEP} = \texttt{anti}(\text{AWAKE})$ et $\text{ASLEEP} = \texttt{V}_0(\text{SLEEP})$.

Afin d'avoir toutes les paraphrases possibles (cf. figure 43 page 59, pour chaque constante $u' = f(u)$ associant une FL $f$ et une UL $u$, j'ai créé dans $\Sigma_{fl}$ une constante $u'$ par valeur possible pour $u$. Si une même UL peut être obtenue à partir de deux couples $(f, u)$ différents, je crée deux constantes $u'_1$ et $u'_2$ pour différencier ces deux cas dans $\Sigma_{fl}$. Alors, dans $\Sigma_{finfl}$, je crée pour chaque LU possible une constante $u''$. Les lexiques $\mathcal{L}_{lexfl}$ et $\mathcal{L}_{reducefl}$ sont tels que :

— $\forall\ (f, u).\ (\forall\ u'.(u' = f(u).\ \mathcal{L}_{lexfl}(u') = f(u)))$

— $\forall\ u'_1\ u'_2.\ (u'_1 \neq u'_2 \wedge \exists\ u''.\ u'' = \mathcal{L}_{reducefl}(u'_1) = \mathcal{L}_{reducefl}(u'_2)) \Rightarrow f_1(u_1) = f_2(u_2)$ où $f_1$, $u_1$, $f_2$, $u_2$ sont tels que $\mathcal{L}_{lexfl}(u'_1) = f_1(u_1)$ et $\mathcal{L}_{lexfl}(u'_2) = f_2(u_2)$

**Problème rencontré :** Le problème qui s'est ici posé, est en lien avec la réversibilité des règles d'équivalence entre les FL (cf. figure 33c page 51). En effet, à partir de l'arbre de gauche de la figure 33e page 51, l'arbre de droite de ce même exemple est obtenu. Cependant, si l'on a l'arbre de droite (sans n'avoir jamais eu l'arbre de gauche), l'arbre de droite n'était pas trouvé. En effet, ACGtk ne savait pas que UPSET était $\texttt{anti}(\text{CALM})$.

Afin de faire apparaitre ce lien, j'ai donc ajouté dans l'ordre des commandes à réaliser pour générer une phrase les deux transductions de $\Sigma_{deep\text{-}rel}$ à $\Sigma_{finfl}$, et de $\Sigma_{finfl}$ à $\Sigma_{deep\text{-}rel}$ avant l'étape de paraphrase syntaxique profonde.

## 6.5 Modélisation de la syntaxe de surface

Cette modélisation est constituée des deux signatures $\Sigma_{surface\text{-}syntactic}$ et $\Sigma_{surface\text{-}tree}$, ainsi que des lexiques $\mathcal{L}_{ssynt}$ et $\mathcal{L}_{stree}$ (cf. partie orangée sur la figure 10 page 15).

### 6.5.1 Objectifs

Le but de cette partie de la modélisation, à savoir la modélisation de la syntaxe de surface, est d'effectuer la transition de la représentation de syntaxe profonde à la syntaxe de surface. À la fin de la modélisation de la syntaxe profonde, les FLs ont été réalisées.

Il s'agit donc ici de construire les arbres de syntaxe de surface à partir des arbres de syntaxe profonde. Comme leurs prédécesseurs, les arbres de syntaxe de surface sont des arbres de dépendance, dont les nœuds sont des lexèmes et dont les branches sont étiquetées par des relations, mais cette fois ce sont des relations de syntaxe de surface.

Cette modélisation doit donc permettre de représenter ces arbres de syntaxe de surface, avec leurs différentes relations de syntaxe de surface sur les branches, et les lexèmes au niveau des nœuds.

### 6.5.2 Implémentation

**Représentation :** La signature $\Sigma_{surface\text{-}tree}$ modélise la représentation de syntaxe de surface. J'y ai défini huit types :

— les types $T$ et $Tb$, qui ont le même rôle et sens qu'en syntaxe profonde,

— le type *rel*, qui a le même rôle et sens qu'en syntaxe profonde, si ce n'est qu'il caractérise les relation syntaxiques de surface ici,
— les types $N$, $V$, $Adj$, $Adv$ et $P$, qui ont pour les quatre premier le même rôle et sens que pour la syntaxe profonde, et le dernier correspond au type des prépositions.

En effet, les prépositions n'étaient pas présentes en syntaxe profonde mais apparaissent ici, en syntaxe de surface.

Les opérateurs $*$ et $\wedge$ ainsi que la constante *void* y sont également définis, et ont le même type et rôle que dans la signature $\Sigma_{deep\text{-}rel}$.

De plus, j'ai de nouveau défini *verb*, *noun*, *adj* et *adv*, et ai défini *prep*, qui fonctionne comme les 4 premiers (cf. section 6.4.2) à cela près que son premier argument est une préposition.

Les relation syntaxiques de surface également définies, telles que *Subj* (pour le sujet), *DirO* (pour le complément d'objet direct), ou encore *OblO* (pour le complément d'objet indirect).

**Exemple 6.5.1** *La phrase* « John assassinate Mary the wife of Alain. » *sera représentée comme en figure 21 au niveau de la signature* $\Sigma_{surface\text{-}tree}$*. L'obtention de cette implémentation est détaillée dans le paragraphe suivant.*

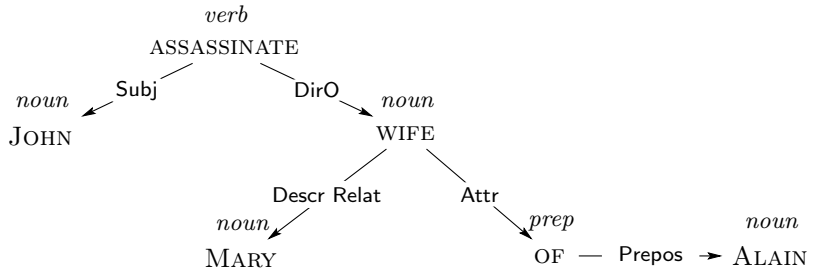



FIGURE 21 – Illustration de la représentation de syntaxe de surface de la phrase « *John assassinate Mary the wife of Alain.* » dans $\Sigma_{surface\text{-}tree}$

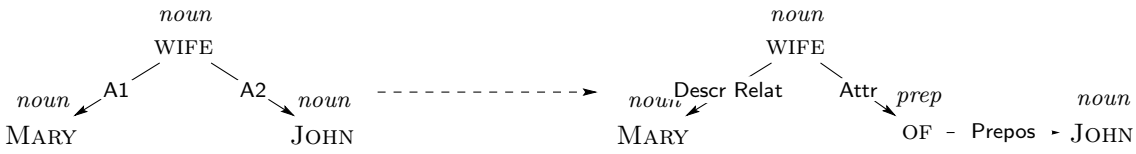
Ainsi, un arbre de syntaxe de surface théorique sera encodé de la manière suivante. Prenons comme exemple la phrase de l'exemple 6.5.1 :

1. La racine de de l'arbre est un lexème, qui doit donc être encodé dans $\Lambda(\Sigma_{surface\text{-}tree})$, avec comme type celui correspondant sa nature grammaticale : $ASSASSINATE : V$;. La modélisation de cet arbre commencera donc par la constante associée à la nature grammaticale. Dans le cas de l'exemple 6.5.1, on aura $verb\ ASSASSINATE$ ....
2. Si ce nœud racine a des fils, chacun de ces fils est un arbre, attaché à la racine par une branche étiquetée d'une relation syntaxique de surface. On attache donc ces arbres fils à la racine par les bonnes relations : $verb\ ASSASSINATE\ ((Subj * fils1) \wedge (DirO * fils2))$ .
3. Il reste à encoder les fils. Pour cela, répéter les étapes 1 et 2 autant de fois que nécessaire en remplaçant "racine" par "nœud père". On obtient finalement :
   $verb\ ASSASSINATE\ ((Subj * (noun\ JOHN\ void)) \wedge (DirO * (noun\ WIFE\ ((Descr_relat * (noun\ MARY\ void)) \wedge (Attr * (prep\ OF\ (Prepos * (noun\ JOHN\ void))))))))$.

On pourra remarquer que la préposition $OF$ a été introduite lors de la transition de la syntaxe profonde à la syntaxe de surface.

**Transition de la syntaxe profonde à la syntaxe de surface :** Cette transition correspond à la transduction de $\Sigma_{finfl}$ à $\Sigma_{surface\text{-}tree}$. Comme on a pu le remarquer précédemment avec le cas de WIFE, certaines structures d'arbres sont à modifier lors de cette transition.

Cependant, certains arguments sémantiques sont optionnellement exprimables (cf. section 6.3), comme c'est le cas de l'UL WIFE. Dans le cas où le premier argument sémantique est omis, le premier actant en syntaxe profonde est représenté par la branche étiquetée par la relation $A1$, au bout de laquelle se trouve le nœud feuille *nil*. On a donc deux arbres différents possibles en syntaxe de profonde, qui doivent donner lieu à deux arbres différents en syntaxe de surface, qui donneront lieu à deux phrases différentes par la suite (cf. figure 22 page suivante).




$noun\ WIFE\ ((A1\ *\ (noun\ MARY\ void))\ \wedge\ (A2\ *\ (noun\ JOHN\ void)))$

$noun\ WIFE\ ((Descr_relat\ *\ (noun\ MARY\ void))\ \wedge\ (Attr\ *\ (prep\ OF\ (Prepos\ *\ (noun\ JOHN\ void)))))$

(a) « *Mary the wife of John* »




$noun\ WIFE\ ((A1\ *\ nil)\ \wedge\ (A2\ *\ (noun\ JOHN\ void)))$

$noun\ WIFE\ (Poss\ *\ (noun\ JOHN\ void))$

(b) « *The wife of John* »

FIGURE 22 – Illustration des arguments optionnellement exprimables dans les représentations en syntaxe profonde (à gauche) dans $\Lambda(\Sigma_{deep\text{-}rel})$ (ou $\Lambda(\Sigma_{finfl})$) et en syntaxe de surface (à droite) dans $\Lambda(\Sigma_{surface\text{-}tree})$, accompagnés des $\lambda$-termes correspondant

**Remarque 6.5.2**

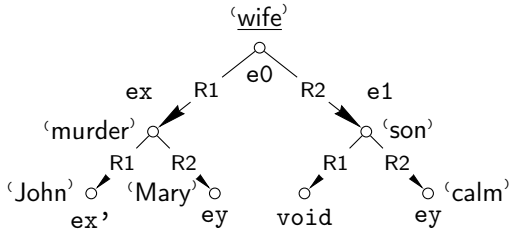
a. *La notion d'argument optionnellement exprimable disparait complètement par transduction (de $\Sigma_{finfl}$ à $\Sigma_{surface\text{-}tree}$) dans cette étape de la modélisation. Si il en restait une trace en syntaxe profonde avec la constante nil, il n'en reste ici aucune trace.*
b. *Dans le second cas, où le sujet n'apparait pas, la phrase* « John's wife » *m'a paru plus naturelle que* « the wife of John ».

Il faut donc traiter ces deux arbres de syntaxe profonde comme deux arbres complètement différents, d'autant plus que l'un utilise la préposition OF et l'autre le possessif. Ils n'ont même pas de sous-arbre en commun. Aussi, la signature $\Sigma_{surface\text{-}syntactic}$ a été nécessaire afin d'identifier les différents patrons propres à chaque lexèmes, au niveau syntaxique profond, avant de de créer grâce au lexique $\mathcal{L}_{stree}$ leur arbre de syntaxe de surface (cf. figure 22 qui illustre deux structures différentes pour WIFE , et la figure 46 page 61 qui présente le code associé). Ce même raisonnement s'applique aux arbres résultants de la paraphrase syntaxique profonde. Les fils d'un nœud pouvant différer, comme c'est le cas de BE qui peut avoir ou non la FL `non` en relation $ATTR$, il faut dans ce cas aussi d'autres patrons (cf. figure 45 page 60).

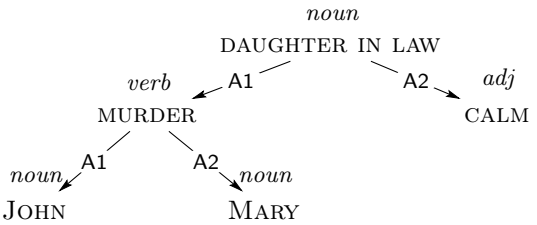
**Signature $\Sigma_{surface\text{-}syntactic}$ :** Cette signature contient donc le vocabulaire associé aux patrons d'arbres ou sous-arbres liés à chaque lexème. En outre, je me suis inspirée de Moortgat 1997 et de Morill 1994, qui donnent des exemples sur les logiques des types catégoriels, pour typer ces patrons. En effet, ils vont contraindre la nature grammaticale des fils de ce patron (cf. exemple 47 page 61), et donc empêcher que certains arbres voués à générer des phrases absurdes voient le jour en syntaxe de surface.

Ainsi, le graphe sémantique de la figure 23 page suivante est correct de par sa construction (même si on voit déjà qu'il ne voudra intuitivement rien dire). Il va donc donner lieu à un arbre de syntaxe profonde, lui aussi correct de par sa construction. Cependant (heureusement), il ne donnera pas lieu à un arbre de syntaxe de surface, car la nature grammaticale des fils de DAUGHTER IN LAW n'est pas conforme à ce qui est spécifié dans $\Sigma_{surface\text{-}syntactic}$ (cf. figure 44 page 60) ! En effet, le fils gauche est un groupe verbal (de type $S$), alors qu'il doit être un groupe nominal (de type $NP$).

**Groupes adverbiaux :** Les groupes adverbiaux, qui avaient un comportement inspiré des TAG au niveau de la modélisation sémantique, et qui, au niveau de la modélisation de la syntaxe profonde, avaient également une représentation inspirée des TAG, se fondent ici dans le décor. En effet, « *John murder Mary for a political reason.* » sera représenté dans $\Sigma_{surface\text{-}tree}$ par le graphe de la figure 24 page suivante.




(a) Représentation d'un graphe sémantique correct mais intuitivement absurde dans $\Lambda(\Sigma_{semantic})$




(b) Représentation d'un arbre de sntaxe profonde correct mais intuitivement absurde dans $\Lambda(\Sigma_{deep\text{-}rel})$

FIGURE 23 – Représentation dans $\Lambda(\Sigma_{semantic})$ et $\Lambda(\Sigma_{deep\text{-}rel})$ d'une même "phrase" qui n'aura pas de suite dans la génération vers la syntaxe de surface

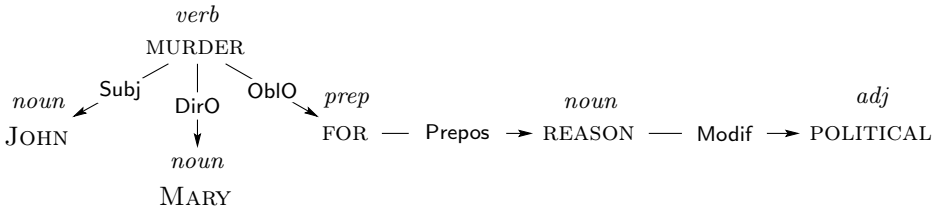



FIGURE 24 – Représentation de la phrase « *John murder Mary for a political reason.* » en syntaxe de surface dans $\Lambda(\Sigma_{surface\text{-}tree})$, dont le $\lambda$-terme correspondant est $verb\ MURDER\ (((Subj\ *\ (noun\ JOHN\ void))\ \wedge\ (DirO\ *\ (noun\ MARY\ void)))\ \wedge\ (OblO * (prep\ FOR\ (Prepos * (noun\ REASON\ (Modif\ *\ (adj\ POLITICAL\ void)))))))$

### 6.5.3 Problèmes rencontrés

Je n'ai pas rencontré de gros problème lors de cette modélisation, seulement un problème de conception lorsque j'ai voulu modéliser la syntaxe de surface. En effet, le premier actant de syntaxe profonde n'est pas toujours le sujet, le second pas toujours le COD, etc. C'est pour cela qu'il y a deux signatures $\Sigma_{surface\text{-}syntactic}$ et $\Sigma_{surface\text{-}tree}$, afin de faire correspondre les arbres de syntaxe profonde à ceux de syntaxe de surface, sur le même principe que les règles de paraphrases, ou les fonctions lexicales, soit sur le principe de la transduction, mais cette fois ce sont des patrons de construction détaillés et non des demi-règles d'équivalence.

# 7 Evaluation

L'évaluation de cette solution est plus qualitative que quantitative. En effet, il s'agit ici surtout de tester si générer du texte à l'aide de la TST et des ACGs est réalisable par un modèle, et donc de vérifier que le texte produit par le modèle en question (ou, ici, la structure syntaxique de surface produite) est compréhensible pour un locuteur natif, et correcte.

## 7.1 Échantillons de test

J'ai donc choisi un ensemble d'exemples tirés des volumes Mel'čuk et al. 2012 et Mel'čuk et al. 2013, qui donnent un très bon aperçu des phénomènes lexicaux que l'on peut rencontrer dans la langue naturelle tels que les collocations, comme des phénomènes propre à la TST, tels que les FLs ou la paraphrase.

Ainsi, les UL ASSASSINATE et DAUGHTER IN LAW mettent en exergue la paraphrase sémantique (cf. exemples 6.3.2 page 18 et 6.3.3 page 18), le groupe adverbial « *for a political reason* » illustre le traitement des groupe adverbiaux, traitement inspiré des TAG, et qui n'est pas le même que celui utilisé pour la plupart des autres UL (cf. figures 16a page 22 et 18 page 23).

Les UL WIFE et SON mettent en lumière les arguments sémantiques optionnellement exprimables, et l'apparition du possessif en syntaxe de surface (cf. exemples 22 page ci-contre). La phrase « *Alain*

*is calm.* » illustre elle tous les aspects de la paraphrase syntaxique profonde, que ça soit les règles de lexicalisation, de restructuration, ou l'utilisation des FLs (cf. annexe D).

J'ai également implémenté une partie des champs lexicaux associés à SLEEP, AWAKE, DOUBT et HOPE en plus de CALM et UPSET, afin d'illustrer la génération ou non des arbres issus de la paraphrase syntaxique en fonction des mots présents dans le lexique (par exemple $\mathtt{V}_0$(CALM) n'existe pas, mais $\mathtt{V}_0$(ASLEEP) = SLEEP : l'arbre de droite de la figure 33a page 51, en remplaçant CALM par SLEEP doit donner la phrase « *Alain sleep.* », alors que dans le cas de CALM, cet arbre ne peut générer aucune phrase, car le terme correspondant à $\mathtt{V}_0$(CALM) n'existe pas).

Enfin, toutes ces UL ont un traitement différent en syntaxe de surface, ce qui permet d'illustrer plusieurs cas possibles de relations syntaxiques de surface, et comment elles apparaissent. Les exemples que j'ai utilisés sont donc restreints, mais relativement riches pour avoir un bon aperçu de ce que ce modèle permet et de ses limitations.

## 7.2 Méthodologie

Aussi, à chaque étape (soit après l'écriture de chaque lexique), je testais différentes phrases avec les commandes propres à ce lexique, que ça soit pour le parsing ou l'analyse de cette ACG. Les phrases que je testais étaient aussi bien des phrases qui ne devaient pas poser de problèmes que des phrases aberrantes, dont la commande devait renvoyer une erreur, ou aucune solutions. Étant amenée à modifier régulièrement des parties de code écrites précédemment, je testais aussi la génération de ces phrases depuis leur graphe sémantique, jusqu'à la partie que j'étais en train d'implémenter. Les commandes pour faire cela sont décrites dans la figure 34 page 53.

J'ai par la suite écrit des scripts de tests pour un certain nombre de phrases possibles, qui contiennent les différentes commandes à executer pour générer, à partir du graphe sémantique, l'arbre de syntaxe de surface. Le code de l'un de ces scripts de tests se trouve en annexe F. Certains de ces scripts de tests contiennent toutes les commandes de la boucle de paraphrase syntaxique profonde (cf. section 6.4.3). Cependant, comme je l'ai expliqué en section 6.4.3, j'ai supposé disposer d'un contrôleur pour executer ces commandes automatiquement, car il peut y en avoir beaucoup, et il est assez compliqué de s'y retrouver à la main (cf. algorithme 1 page 27 et annexe E). Parmi ces scripts de tests, certains traitent également des exemples qui ne doivent pas aboutir : pendant l'exécution des commandes, soit une erreur doit survenir, soit, lors d'un parsing, ACGtk doit ne pas trouver de solution, ce qui arrête la génération.

## 7.3 Perspectives

Une évaluation future possible de ce modèle est sa capacité de généralisation. En effet, tous les résultats obtenus sont positifs, les arbres de syntaxe de surface obtenus sont corrects, et les représentations qui échouent lors de la génération auraient donné lieu à des phrases aberrantes. Il faudrait donc maintenant étoffer les lexiques utilisés en vocabulaire, mais également en FLs (le nombre de FLs que j'ai manipulées est assez restreint), pour tester sa capacité de généralisation.

Cette dernière pourrait être testée selon son application au RLF, en comptant le nombre de nœuds et le nombre de fonctions lexicales du RLF couvert(e)s par ce modèle par rapport à leur nombre total dans le RLF.

# 8 Analyse des résultats

Je vous ai présenté mes résultats dans la section précédente ; cette section porte un regard critique sur ceux-ci, où je fais un bilan des résultats obtenus vis-à-vis des objectifs initiaux, pour aborder finalement en sous-section 8.2 les limitations de ce projet.

## 8.1 Bilan

Les objectifs de ce stage (cf. section 3.2.1) étaient :

1. d'étudier les structures linguistiques proposées par la TST pour les différents niveaux de représentation de la langue et les modules de transition associés,

2. d'utiliser les approches et outils développées par Sémagramme afin de modéliser la TST (au moins jusqu'au niveau de syntaxe de surface ou au niveau morphologique profond), d'implémenter et de tester le modèle produit,
3. de formaliser et spécifier les transformations, de prendre en compte les ressources lexicales nécessaires, et de modéliser des collocations,
4. d'exploiter les propriétés des ACGs pour réaliser l'objectif 3 entre autres,
5. d'obtenir un modèle fonctionnant sur des exemples non triviaux couvrant un certain panel des expressions linguistiques,soit un modèle contenant des exemples de collocations, de FLs, et des exemples permettant la paraphrase au cours de leur génération.

Tous ces objectifs ont été remplis. En effet, après avoir passé un peu plus de deux mois à étudier la TST, (Polguère 2003), ce qui avait déjà été fait dans le cadre de la génération de texte avec la TST, et comment utiliser ACGtk, j'ai travaillé sur la réalisation de ce modèle, en implémentant au fur et à mesure mes idées dans ACGtk afin de les tester. Comme décrit en section 7, j'ai traité des exemples de collocations, des exemples utilisant la paraphrase sémantique, des exemples utilisant la paraphrase syntaxique profonde, et des exemples utilisant les FLs. De plus, comme décrit en section 6, j'ai formalisé et spécifié toutes les transformations ayant lieu lors de la génération de ces exemples dans ACGtk. Aussi, les trois niveaux de représentation que j'ai modélisés sont représentés chacun par une signature ($\Sigma_{semantic}$, $\Sigma_{deep\text{-}rel}$ et $\Sigma_{surface\text{-}tree}$), et les modules de transition de la TST sont également modélisés dans cette implémentation (cf. figure 10 page 15).

Cependant, je pensais initialement utiliser Grew en plus d'ACGtk, mais cela n'a pas été le cas. De plus, comme j'ai été confrontée lors de la modélisation à une multitude de petits problèmes de conception et d'implémentation (comme pour les deux types de paraphrases, la représentation des groupes adverbiaux, le traitement des arguments optionnellement exprimables, etc.), je me suis focalisée sur de petits échantillons de test et je n'ai pas généralisé ce modèle. Je n'ai donc pas de test généralisé ou de grande couverture du RLF par exemple.

Néanmoins, les résultats obtenus sont corrects (du point de vue de leur structure et de leur représentation), justes (dans le sens de "attendus") et satisfaisants pour les phrases en question. Toutes les paraphrases générées sont correctes, c'est à dire qu'aucune phrases incorrectes n'atteint la syntaxe de surface lors de la génération. J'ai traité des phrases ni trop simples ni trop complexes, ce qui m'a permis d'avoir un bon échantillon des phénomènes lexicaux possibles (FLs, paraphrase). Pour chacun de ces phénomènes, j'ai traité un ou plusieurs exemples. Tous ces exemples ont eu le comportement voulu, et ont généré les représentations voulues. J'estime donc avoir une très bonne couverture de test, sur mon échantillon restreint d'exemples, et couvrir également un bon nombre des phénomènes lexicaux les plus fréquents.

## 8.2 Limitations et perspectives

### 8.2.1 Limitations

Les échantillons de tests de ce projet ne permettent, au niveau du vocabulaire et des FLs implémentées, que de former quelques phrases. Le nombre de phrases générables est donc limité. De plus, aucune étude statistique sur le bon traitement de certains phénomènes lexicaux n'est possible, ces phénomènes étant représentés que trop peu de fois dans ce modèle.

Comme je l'ai expliqué en section 6.2, je n'ai pas représenté la structure communicative. Cela a pour conséquence que j'ai désigné arbitrairement par mon implémentation quel est le nœud dominant d'un graphe sémantique et comment la racine d'un arbre de dépendance de syntaxe profonde est choisi.

Concernant mon implémentation du niveau de représentation sémantique, il y a quelques défauts. En particulier, on ne peut pas en semantique accéder à un nœud qui n'est pas la racine d'un sous arbre (cf. figure 15 page 21). Mes lectures sur la TST me laissent penser que cela devrait être le cas, je n'ai cependant pas trouvé ici de modélisation parfaite, et ai préférer avancer avec cette modélisation convenable. De plus, selon comment on interprète l'étape de paraphrase sémantique, on peut comprendre que soit chacun des graphes obtenus en "dépliant" les définitions peut donner lieu (sous conditions) à un arbre de dépendance de syntaxe profonde, soit que seule la définition irréductible (que l'on ne peut pas "déplier" plus) doit être considérée comme paraphrase. Mon implémentation réalise cette deuxième option. En effet, la complexité des liens entre signatures et le nombre de ces signatures aurait considérablement augmenté si j'avais implémenté la première option. J'ai donc choisi la seconde option, qui, si ce n'est peut-être pas celle à

privilégier, reste un bon compromis de la première option.

Comme évoqué en section 6.4.3 et 8.1, j'ai supposé qu'il y a un contrôleur pour executer les commandes et contrôler la boucle de paraphrase syntaxique, avec les contraintes nécessaires pour l'arrêter, sans avoir toutefois implémenté ce contrôleur. C'est cependant un mécanisme que l'on sait implémenter, et qui n'est pas le but premier du stage.

De plus, sauvegarder chaque nouvel arbre qui apparait lors de la paraphrase de syntaxe profonde n'est pas très adapté aux ACGs, car cela nécessite un contrôle extérieur, et d'autres structures de données pour stocker ces arbres et aller lire dans cet espace de stockage.

### 8.2.2 Perspectives

Cependant, j'ai traité des phrases simples et légèrement complexes, avec un exemple de chaque phénomène voulu (adverbes, fl, paraphrase), qui permettent d'être optimiste quant à la généralisation.

Si je n'ai pas traité directement de collocation, j'ai utilisé et travaillé sur les FLs, or les collocations sont une application des FLs. Cela est donc très facilement faisable avec mon modèle.

Tous les cas particuliers que sont mes exemples montrent de plus que le phénomène lexical qu'ils représentent est implémentable dans mon modèle, et que ce modèle peut être généralisé à toutes ces familles de champ lexicaux.

De plus, la transition de la représentation sémantique à la représentation de syntaxe profonde (le transduction de $\Sigma_{semantic}$ à $\Sigma_{deep\text{-}rel}$) a très bien fonctionné, et est très bien adapté aux ACGs. Cette transition est d'ailleurs assez complète, car j'ai pu y rajouter la modélisation des arguments sémantiques optionnellement exprimables. Les ACGs sont également très bien adaptées à la réalisation des FLs (qui correspond à la transduction de $\Sigma_{deep\text{-}rel}$ à $\Sigma_{finfl}$), et à la transition de la représentation de syntaxe profonde à celle de syntaxe de surface (soit la transduction de $\Sigma_{finfl}$ à $\Sigma_{surface\text{-}tree}$). En effet, les bonnes structures sont générées, et celles qui sont incorrectes ne le sont pas, celles qui ne doivent pas aboutir n'aboutissent pas. Réciproquement, tous les antécédents possibles de ces structures sont effectivement trouvés, et si la structure est incorrecte, aucun antécédent ne lui est trouvé. Néanmoins, la partie traitant de la paraphrase syntaxique profonde n'est pas très bien adaptée aux ACGs, comme le laisse entendre les sections 6.4.3 et 8.1, et comme l'explique la sous-section précédente.

Les deux principales perspectives possibles de ce travail sont d'écrire le contrôleur de la paraphrase de syntaxe profonde (ou modéliser cette dernière autrement), et généraliser ce modèle.

# 9 Avancement

Mon travail s'est divisé en trois grandes parties, à savoir l'étude bibliographique, la modélisation et l'implémentation des différentes étapes de génération, et l'écriture des scripts de tests et de ce présent rapport.

**Méthodologie :** En ce qui concerne l'étude bibliographique, j'ai pris des notes de chacune de mes lectures, en les référençant d'une part sur une liste papier triée par catégorie (TST, ACG et autre) afin de retrouver rapidement dans quelles notes chercher quelles informations, et d'autre part en les référençant dans un fichier .bib afin de ne pas en oublier dans ce rapport, mais également d'avoir des références complètes, ou d'avoir le temps de chercher les informations manquantes pour qu'elles soient complètes. À peu près aux deux tiers de cette étude, j'ai présenté la TST et ce que j'avais appris dessus à l'équipe Sémagramme, ce qui m'a permis d'organiser ce que j'avais appris, et qui m'a été bien utile pour la suite.

Lors de la modélisation et de l'implémentation, je testais sur papier mes idées de modélisation avant de les implémenter, de les déboguer, puis de les tester (et de noter dans un fichier texte les commandes de test effectuées). J'ai d'abord écrit l'ACG comportant $\Sigma_{semantic}$, $\Sigma_{deep\text{-}syntactic}$ et $\mathcal{L}_{sem}$, afin de représenter les graphes sémantiques. Je me suis après attelée, au fur et à mesure que les étapes précédentes fonctionnaient comme voulu, à la représentation de syntaxe profonde, à la paraphrase syntaxique profonde, à la réalisation des FLs, puis à la représentation de syntaxe de surface, tout en améliorant le modèle déjà implémenté et en le complétant (avec les arguments sémantiques optionnellement exprimables par exemple).

Pour les scripts de tests et le rapport, j'ai divisé le temps restant entre les deux, de manière à finir la première version de ce rapport à temps pour avoir des retours de mes deux tuteurs, et pour avoir fini les scripts de tests avant la fin du stage. Concernant les scripts de test, j'ai repris les commandes que j'avais déjà effectuées, et les ai organisées et complétées afin de couvrir le plus de cas possible.

**Tâches à effectuer :** Au cours de ce stage, les différentes tâches que j'ai effectuées sont :

1. lecture et prise de connaissance sur les ACGs, sur la TST, et sur ACGtk,
2. faire un état de l'art (sur les ACGs et la TST),
3. écriture d'exemples simple pour la TST, pour les implémenter plus tard avec les ACGs,
4. faire une présentation de la TST à l'équipe Sémagramme,
5. écriture du pré-rapport,
6. écriture et étude d'ACGs simples pour la prise en main
7. établir une méthode théorique ou un modèle pour encoder un MST, au moins en partie (qui utiliserait les ACGs, pouvant comporter des parties non implémentées dans ces deux formalismes potentiellement), pour les parties correspondant à
   (a) la modélisation sémantique,
   (b) la modélisation de syntaxe profonde,
   (c) la modélisation de la paraphrase syntaxique profonde,
   (d) la modélisation de la réalisation des FLs,
   (e) la modélisation de syntaxe de surface,
8. implémentation et mise en pratique de ce modèle pour les parties correspondant aux modélisations 7a, 7b, 7c,7d et 7e,
9. permettre à certains arguments sémantiques d'être optionnellement exprimables et les modéliser dans le modèle déjà implémenté,
10. modéliser les groupes adverbiaux afin que leurs représentations sémantique comme syntaxiques soient correctes, sans ouvrir la porte à des structures incorrectes pour autant,
11. écrire des scripts de test,
12. rédaction du rapport.

Concernant les articles et livres que j'ai pu lire, vous en trouverez la plupart en références bibliographiques de cet article. J'en ai lu certains autres, mais qui parlent plus d'exemples d'implémentation du module sémantique et du module syntaxique profond de la TST, que je n'ai pas cité dans ce rapport (comme (Lareau et al. 2018) par exemple).

Le diagramme de Gantt montrant l'avancement de mon stage en comparant le planning prévisionnel effectué au bout de 6 semaines de stage à celui effectif se trouve en annexe G.

Globalement, j'ai été plus rapide que ce que j'avais prévu, et je n'avais pas anticipé que j'allais devoir revenir sur ce que j'avais déjà implémenté, afin d'y ajouter les arguments sémantiques optionnellement exprimables et d'implémenter correctement les groupes adverbiaux.

# 10 Impressions

Cette partie aborde dans un premier temps mes impressions sur mon environnement de travail, puis aborde ce que ce stage m'a apporté.

## 10.1 Cadre de travail et journée type

J'ai été accueillie le premier jour par mon tuteur Sylvain Pogodalla, qui m'a fait visiter les locaux et m'a présenté mon sujet de stage plus en détail. Dès le lendemain, j'ai été dans le bureau commun aux doctorants, où j'allais effectuer le reste de mon stage. Je travaille donc avec les doctorants et un ingénieur de recherche. L'ambiance y est très agréable, bienveillante et chaleureuse, propice au travail. Le bureau est grand et lumineux, et proche des autres bureaux de l'équipe. De plus, mon statut de stagiaire me permet de bénéficier de tarifs réduits au restaurant du laboratoire, et mon badge ouvre les portes des bâtiments. Je peux donc travailler avec des horaires souples, et manger sur place sans problème. Aussi, j'ai plusieurs occasions quotidiennes de discuter avec les autres membres de l'équipe de leurs travaux, que ce soit les doctorants ou les membres permanents, ce qui est très intéressant, passionnant et convivial.

Lors d'une journée type, j'arrive le matin entre 9h et 9h30, travaille jusqu'à la pause de midi, où tous les membres de l'équipe mangeant à la cantine ensemble. Je travaille encore toute l'après midi jusqu'à 17h30 à peu près. Comme je peux télétravailler sans problèmes, il m'arrive régulièrement de rentrer en milieu d'après midi pour aller finir ma journée de travail chez moi, afin de faire une pause et un peu d'exercice. C'est une coupure appréciable dans la journée.

### 10.2 Apports sur le plan professionnel, technique et personnel

Professionnellement et techniquement parlant, ce stage m'a apporté beaucoup car il m'a fait découvrir le domaine de la linguistique et du TALN. J'ai également approfondi les quelques bases que j'avais en $\lambda$-calcul, et ai pu en apprendre beaucoup plus sur les logiques des types catégoriels. Ce stage m'a beaucoup apporté en terme de connaissances techniques théoriques.

Si j'étais déjà autonome vis-à-vis de ma gestion du travail, j'ai appris à mieux organiser mon temps de travail, à varier les tâches à faire lorsque c'était possible (lors de la recherche bibliographique, la prise de note permettait de reposer mes yeux de la lecture par exemple, car je rédigeais des paragraphes qui résumaient l'article, ou quelques pages d'un chapitre). De plus, si j'étais très bien encadrée, j'ai le sentiment que j'étais libre d'organiser mon travail (les recherches bibliographiques, l'implémentation, les tests, etc.) presque comme je voulais, ce qui m'a également apporté en terme de gestion de projet.

Je n'avais auparavant jamais fait de recherche bibliographique si approfondie et si longue. C'est donc un exercice que j'ai découvert, et si il est long et fastidieux, j'y ai pris goût, car les sujets sue lesquels je lisais des articles ou ouvrages me plaisaient. J'ai beaucoup appris en lisant, et j'ai apprécié cette possibilité de toujours apprendre et de découvrir des approches nouvelles.

De plus, je n'ai trouvé aucune source parlant d'une implémentation de la TST à l'aide des ACGs pour générer du texte, j'ai donc élaboré ma solution à partir de peu. Mes seules bases étaient les aspects théoriques de la TST (principalement (Mel'čuk et al. 2012 ; Mel'čuk et al. 2013 ; Mel'čuk et al. 2015 ; Milićević 2006 ; Nasr 1996)) et des ACGs (principalement (de Groote 2001)), ainsi que des exemples d'implémentation qui ont été faits dans ces deux domaines (notamment (Pogodalla 2017a) et (Pogodalla 2017b) pour les ACGs et (Wanner et al. 2010) et (Lareau et al. 2018) pour la TST). Le fait de concevoir un modèle théorique, de l'implémenter, de corriger les erreurs de conceptions vis-à-vis de l'implémentation, de le tester, de confronter les résultats obtenus evc ceux attendus était aussi très enrichissant. En effet, mes premiers résultats n'étaient pas du tout conformes à ce que j'espérais, et j'ai dû revoir ma conception, et réécrire complètement certaines signatures, voire certaines ACGs. Le fait de voir directement l'impact d'une modélisation sur une implémentation et l'impact de cette implémentation sur les résultats, et ce sur tout un projet m'a beaucoup apporté.

J'avais découvert lors du projet "Introduction à la Recherche en Laboratoire" que les travail dans la recherche me plaisait, et ce stage a confirmé mon attirance pour la recherche. Ce travail a été passionnant, c'est pourquoi j'ai candidaté à une offre de thèse dans la même équipe, que j'ai obtenue.

Je poursuivrai donc mes études par un doctorat, afin de pouvoir continuer à faire de la recherche, mais également afin d'enseigner, ce à quoi j'aspire depuis longtemps.

Sur un plan plus personnel, mon sujet étant très riche et pluridisciplinaire, il m'a énormément plu et a été très stimulant, et m'a permis de m'épanouir dans ce que je faisais. J'y ai également rencontré des personnes formidables qui ont fait que ce stage s'est très bien déroulé ; je me levais le matin en ayant envie d'aller travailler, ce qui est un sentiment que je recherchais vis-à-vis de ma vie professionnelle, je l'ai trouvé ici, et suis très contente de poursuivre en thèse dans cette même équipe.

## 11 Impact environnemental et sociétal

### 11.1 Impact environnemental personnel de votre PFE

**Cadre :** La politique du LORIA (et de l'Inria) est de ne pas prêter de machine aux stagiaires, sauf dans des cas spécifiques (si le stagiaire n'a pas de machine personnelle par exemple, cela doit être possible je pense). Ainsi, dès que je parle de mon travail sur ordinateur, il s'agit de mon ordinateur personnel, qui consomme 2,24 g de $CO_2$ par heure travaillée (calculs effectués d'après cette page internet). Je n'ai pas d'écran supplémentaire, et même si j'ai une orchidée sur mon bureau au LORIA (et un clusia sur mon bureau chez moi), je doute que ces deux petites plantes compensent d'une quelconque manière ma

consommation carbone. Je ne les ai donc pas comprises dans ce calcul. De plus, je ne travaillais pas en ligne la plupart du temps, mais en local sur ma machine, sans avoir besoin d'une connexion internet. Je ne consommais donc pas plus ni moins en télétravaillant qu'en travaillant au bureau.

**Consommation énergétique pendant le travail :** Les deux premiers mois de stage, j'ai principalement lu et étudié des articles ou livres, et préparé une présentation sur la théorie sens-texte à l'équipe Sémagramme, sur mon ordinateur personnel lorsque je travaillais sur un ordinateur. Les livres que j'ai lu étant déjà imprimés, et disponibles pour tous les membres du LORIA, j'ai négligé la consommation carbone lié à leur impression. J'ai imprimé certains articles, et la consommation carbone liée à l'impression de ces articles sur toute la durée de ce stage est de 1400 g eq. $C0_2$ (calculé grâce au site des impressions de l'Inria). Durant ces deux premiers mois, j'ai passé environ 70 heures à travailler sur mon ordinateur, ce qui équivaut à une consommation d'environ 156,8 g de $C0_2$.

Les mois suivants, soit 90 jours, j'ai passé environ autant d'heure sur machine qu'à travailler sur papier, avec une consommation moyenne de 1 page par jour. Travaillant en moyenne 7 heures par jour, cela représente 315 heures sur machine et 45 pages. Cela fait donc 705,6 g + 323 g = 1028,6 g eq. $C0_2$ consommés.

Ma consommation énergétique en équivalent $C0_2$ consommée pendant le travail est donc de 1400 + 156,8 + 1028 = 2584,6 g eq. $C0_2$.

**Impact des déplacements :** Pour aller travailler, j'effectue chaque jour 10 minutes en tram, puis 10 minutes à pieds, pour aller sur mon lieu de stage, comme pour en revenir. En mesurant le tronçon de tram sur Google Maps, j'effectue environ 1 kilomètre en tram par trajet, soit 2 kilomètres par jour. Si j'assimile le tram (plein, car je le prends aux horaires de pointe) à un train, ce site internet indique que la consommation d'une personne pour 1 km est de 5,60g de $CO_2$. Ainsi, je consomme pour mes trajets quotidiens environ 11,2g de $CO_2$ par jour. Sachant que mon PFE dure 127 jours, mes transports ont consommé en tout 1422,4g de $CO_2$.

**Impacts autres :** Par ailleurs, je mange 5 jours sur 5 au restaurant de l'Inria, donc sur les 127 jours de mon stage, cela fait 171 450 g de $CO_2$ en considérant environ 1350g de $CO_2$ consommé pour un repas (calculs effectués d'après cette page internet).

**Bilan :** En tout, j'estime l'impact environnemental personnel de mon PFE à 4006,8 g eq. $CO_2$ consommé, soit un peu plus de 4 kg eq. $CO_2$ consommé. Cela est relativement peu comparé à quelqu'un qui aurait fait un PFE entier de programmation avec des réseaux neuronaux tournant sur plusieurs serveurs.

## 11.2 Impact global du projet

### 11.2.1 Impact environnemental du projet

**Conception :** Ce projet a consisté en l'implémentation d'un modèle très réduit de génération de texte à l'aide de la TST et des ACGs. Aucune fabrication matérielle n'est prévue, ce projet est stocké sur un dépôt git de 1 Mb. La consommation liée à la conception de ce modèle est donc relativement faible, tout le code ayant été écrit hors ligne, puis envoyé vers ce dépôt git via des commits que j'effectuais une à deux fois par jour où je travaillais.

**Utilisation actuelle :** Cette implémentation est prévue pour être utilisée hors ligne. Il faut donc télécharger ACGtk (qui fait 12 Mb et nécessite d'avoir téléchargé Opam au préalable) puis mon dépôt git (ces deux étapes nécessitent une connexion internet), et tout est prêt pour utiliser ce modèle (sans connexion internet). Il n'y a pas de restriction particulière du point de vue de la machine à utiliser. Ce modèle fonctionnera sur toute machine linux à jour (pour des raisons de contraintes de dépendances entre les packages d'Opam, si la machine n'est pas à jour je ne suis pas sûre que le téléchargement se fera sans heurt). La nouvelle version d'Opam est prévue pour être supportée sous Windows, aussi je ne pense pas qu'une distribution Windows pose problème. Il faudra seulement attendre cette nouvelle version de Opam. Ainsi, la préparation du poste d'utilisation est assez simple, compatible avec une grande majorité d'ordinateurs, surtout si ce sont des informaticiens (souvent plus prompts à utiliser Linux que

Windows) qui l'utilisent. Il s'agit alors d'un téléchargement de logiciel similaire à n'importe quel autre téléchargement : cela va consommer des données et de l'électricité, certes, mais reste dans la moyenne de consommation actuelle lors du téléchargement d'un logiciel (je ne dis pas que cette moyenne est bonne, juste que je ne fais pas pire).

Ensuite, en ce qui concerne l'utilisation en local, hors ligne, la consommation de la machine va évidemment dépendre de sa puissance. Cependant, de ce que j'ai pu remarquer, la compilation des fichiers sources prend 0,09 secondes environ sur ma machine, sans que je ne remarque une différence notoire dans l'utilisation de ma mémoire vive ou du CPU avec l'outil *System monitor*. Il faut après lancer le logiciel ACGtk, et exécuter les commandes. Là aussi, en observant l'outil *System monitor*, je ne vois aucune variation significative dans les courbes d'utilisation de mon CPU ou de ma mémoire vive. De plus, les commandes que j'effectue prennent au plus 0,001 secondes : elles sont donc rapides. J'estime donc que la consommation liée à l'utilisation actuelle de ce projet est relativement faible, donc assez bonne.

**Utilisation future potentielle :** Plusieurs cas sont possibles concernant l'utilisation future de ce modèle. J'en distingue quatre en particulier :

1. Ce modèle tombe à l'abandon, et reste à l'étape de "test", pour savoir si les ACGs sont un bon moyen de générer du texte avec la TST. Si ce scénario venait à arriver, très peu de personnes utiliseraient ce modèle, et donc sa consommation énergétique sera quasiment nulle. Je ne pense pas (et n'espère pas) que ce scénario sera le bon.
2. Ce modèle est repris pour être élargis à l'anglais sur un vocabulaire beaucoup plus large, afin de l'améliorer et tester de nouvelles fonctionnalités, implémenter le traitement d'autres phénomènes lexicaux, ou prendre en compte les structures communicatives par exemple. Ce scénario est je pense beaucoup plus probable. Alors, l'équipe de chercheurs travaillant dessus sera amenée à le télécharger, l'utiliser, et y contribuer. Seulement, ce sera un petit nombre d'utilisateurs qui le téléchargeront, et l'utilisation ayant lieu exclusivement hors ligne, la contribution presque exclusivement hors ligne (à l'exception des commits), la consommation qui en découlera sera un plus élevée que celle liée à sa conception et son utilisation actuelle, et selon ce qui est visé, elle pourra être un peu plus importante ou beaucoup plus importante. En effet, en fonction de la taille du vocabulaire implémenté, différentes options sont possibles :
   - Implémenter tout le vocabulaire manuellement : le temps de compilation augmentera drastiquement, et celui d'exécution aussi ; la compilation et l'execution consommeront donc beaucoup plus.
   - Aller lire les données nécessaire à l'élaboration du vocabulaire dans une ressource lexicale adaptée, et enregistrer ces données sous le bon format dans le modèle. Alors, la taille des fichiers sera beaucoup plus faible, la compilation sera donc surement plus rapide que dans le cas précédent, même si l'exécution devrait rester la même. Cependant, une connexion internet sera alors nécessaire.

   Je pense qu'en fonction de la taille du vocabulaire voulu et de l'existence ou non d'une telle ressource lexicale, l'une ou l'autre de ces deux solutions sera meilleure d'un point de vue de la consommation. Quoi qu'il en soit, le nombre de personnes qui utilisera ce modèle (une centaine peut-être) sera assez faible, donc sa consommation ne sera, somme toute, pas si grande.
3. Ce modèle est repris dans le but de générer du texte dans un contexte multilingue (ce que la TST est réputée pour faire très bien). Il faut alors un vocabulaire dans chacune des langues concernées. La consommation de ce scénario sera donc au moins celle du scénario précédent multipliée par le nombre de langue utilisées. De plus, un nombre d'utilisateurs potentiellement plus grand sera concerné, ce qui multiplie encore la consommation de ce modèle. Cependant, cette consommation sera équivalente à celle d'un logiciel qui existe en plusieurs langues et qui nécessite une connexion internet par exemple. Par exemple, Facebook remplit ces critères mais utilise beaucoup d'autres données que de simples dictionnaires : la consommation de ce logiciel restera bien moins inférieure à celle de Facebook, tout en étant utilisée par drastiquement moins d'utilisateurs (je doute que des millions d'utilisateurs utilisent ce modèle à des fins de recherche... Je pencherais plutôt pour une ou plusieurs dizaines de personnes tout au plus).
4. Ce modèle a été développé dans un contexte multilingue (scénario 3), et remplit toutes les fonctionnalités de la langue naturelle voulues dans chacune des langues (scénario 2, autant de fois qu'il

y a de langue concernée), et est un succès énorme : il sera utilisé dans d'autres logiciels ou applications, et pourra même (c'est très utopique) remplacer Google Traduction. La consommation de ce modèle va exploser par rapport à ce qu'elle était avant, mais, étant utilisé au sein d'autres logiciels qui utiliseraient sa capacité à générer du texte multilingue, sa consommation resterait tout de même dans la moyenne des consommations logicielles je pense, même si utilisé par des millions de personnes.

De plus, il est probable que ce travail inspire d'autres travaux, à consommation peut être meilleure, ou peut-être pire (avec des réseaux de neurones par exemple). Je ne sais donc pas quel sera l'impact environnemental futur de ce projet, mais j'espère qu'il ne sera pas mauvais, ou au moins raisonnable, et qu'il ne donnera pas naissance à des projets énergivores.

Ce qui est, je trouve, plus probable, est que seules des personnes intéressées par la linguistique ou la génération de texte l'utiliseront, et cela réduit grandement le nombre d'utilisateurs potentiels, car seuls les linguistes ou les chercheurs et ingénieurs en TALN seront concernés.

Si mon projet est continué à grande échelle, les utilisateurs potentiels pourront être internationaux, mais leur nombre sera toujours restreint je pense.

### 11.2.2 Impact sociétal du projet

**Impact sociétal direct :** Mon projet s'intéressant aux langues et utilisant comme ressources les langues elles-mêmes, il ne pose aucun problème direct à la protection de la vie privée. Si toutefois ce projet venait à être utilisé pour la génération de texte à partir de données personnelles (comme l'est le réalisateur MARQUIS à partir de données météorologiques par exemple (Wanner et al. 2010)), il faudrait alors se demander si la règlementation RGPD est respectée. Cependant, les données lexicales étant traitées parfaitement objectivement, je ne vois (à l'heure actuelle) par comment des données privées pourraient rester privées. Je pense que dans l'état actuel du projet en tout cas, ces données seraient publiques et traitées toutes de la même manière. Aussi, si le sexisme ou le racisme est proscrit (mis à part les usages des langues, comme la prédominance du masculin lorsqu'il y a plusieurs personnes concernées, d'un point de vue grammatical en français), la vie privée serait clairement compromise. En effet, la génération de texte à l'aide de la TST est basée uniquement sur des règles de grammaire langagières. Dans la décision des paraphrases utilisées par exemple, aucune discrimination autre que celles impliquée par la grammaire de la langue concernée n'a lieu.

En outre, ce projet n'a pas pour effet de renforcer la dépendance des utilisateurs aux outils numériques et une utilisation accrue de ces outils numériques, car il vise à améliorer ou remplacer si cela est possible des algorithmes déjà existants. Il n'a pas non plus pour effet de renforcer la dépendance au numérique, pour la même raison que précédemment.

**Impact sociétal indirect :** L'un des buts de la TST est d'aider les locuteurs à s'exprimer correctement (i.e. à dire ce qu'ils souhaitent de la bonne manière, avec les bons termes) dans la langue de leur choix. Cela pourrait donc rapprocher des personnes parlant deux langues différentes, et aider au partage de cultures. De plus, certaines langues, comme certains dialectes en Afrique sont très peu parlés, mais ont tout de même une structure grammaticale interne. Intégrer ces langues à ce modèle permettrait d'échanger beaucoup plus facilement avec ces personnes.

D'un point de vu plus proche de l'Histoire et des traditions, si l'on référence tous les dialectes en perdition dans ce modèle, cela permettra d'en garder une trace fiable et de les comprendre même s'ils ne seront plus parlés et auront disparu.

## 11.3 Politique de la structure d'accueil

**Petits gestes quotidiens :** Le LORIA partage ses locaux avec l'Inria, à Vandœuvre-les-Nancy. Au sein de ces locaux, une politique de tri des poubelles est mise en place. En effet, il y a dans chaque pièce une poubelle spécifique au papier, et une poubelle normale pour les autres déchets. Dans les bâtiments, on retrouve des bacs spécifiques pour le recyclage et pour le tri des piles usagées.

**L'Inria :** Au sein de l'Inria, il y a des équipes qui travaillent sur des sujets de recherche axés sur l'environnement (Berthoud et al. 2019), mais très peu au prorata des équipes. De plus, l'Inria a créé un poste d'adjoint au directeur scientifique spécifiquement en charge des questions environnementales, ainsi

qu'une mission de chef de projet pour la mise en place d'une politique de Responsabilité Sociétale et Environnementale (RSE). Aussi, l'Inria présente annuellement à son Conseil d'administration les actions mises en œuvre pour réduire son empreinte carbone.

**Le LORIA :** Le LORIA a quand à lui, en commun avec l'Inria de Nancy, une commission pour l'action et la responsabilité écologique (CARE) qui pointe du doigt l'obligation de l'Inria, du CNRS et de l'UL (Université de Lorraine) de respecter l'accord de Paris sur le climat, mais qui indique aussi que la recherche, de par les déplacements des chercheurs pour se rendre en conférence a un impact très négatif et une forte consommation carbone. Cette association travaille avec la cheffe de projet pour la mise en place d'une politique RSE de l'Inria. Parmi les mesures mises en place, il y a le tri sélectif comme indiqué plus haut, la mise en place de limites de chauffage ou de refroidissement, l'amélioration de l'isolation des locaux, et la mise en place d'un repas végétarien par jour par exemple. De plus, ils essayent d'inciter les chercheurs à voyager que lorsque cela est nécessaire et de privilégier le train à d'autres moyens de transports plus polluants (comme l'avion par exemple). La CARE organise également différents ateliers de sensibilisation, comme par exemple un atelier compost, lors duquel ils ont présenté toutes les solutions possibles pour faire du compost en maison, appartement, avec ou sans jardin, et un atelier pour fabriquer de la lessive à base de cendres. Je n'ai reçu les mails de communication seulement sur ces deux ateliers, et n'ai pu me rendre qu'au premier. Il n'y avait que peu de personnes présentes (moins d'une vingtaine je pense) par rapport au nombre de personnes travaillant dans les locaux, je trouve cela dommage.

**Réflexions personnelles :** Si la recherche n'est pas une entreprise qui va avoir un service de production et qui va polluer de par une chaîne de production, elle a tout de même un impact à réduire. Concernant la politique de l'Inria concernant l'environnement, à part pour le tri sélectif, j'ai dû chercher en ligne pour savoir quelle était cette politique, et je regrette qu'il n'y ait pas plus de communication à ce sujet. En outre, mes collègues de bureau n'avaient pas non plus de réponse immédiate à ma question quand je leur ai demandé si il savaient quelle était cette politique, et je n'ai pas trouvé de chiffres montrant l'évolution récente de la consommation carbone du LORIA, je me pose donc des questions quant à l'efficacité et la réalisation de cette politique. En effet, les petites actions du quotidien ont bien lieu, comme le repas végétarien, le tri sélectif, etc., mais les plus grandes actions telles qu'inciter les chercheurs à voyager moins et à voyager plus écologiquement, je ne sais pas. La crise sanitaire du COVID a aussi dû perturber cela, ce qui pourrait expliquer que je n'ai pas trouvé beaucoup d'informations à ce sujet.

# 12 Conclusion

La TST est une théorie linguistique qui se base sur sept niveaux de représentation et sur six modules de transition, qui utilisent la paraphrase comme concept central. Les fonctions lexicales sont amenées à jouer un role important lors de la génération de texte à l'aide de cette théorie. Les ACGs sont un formalisme grammatical permettant l'encodage d'autres formalismes grammaticaux et entre autre la transduction entre deux langages objets. J'ai modélisé et implémenté au cours de stage un modèle à l'aide du logiciel ACGtk avec des ACGs pour les trois niveaux de représentation sémantique, syntaxique profond et syntaxique de surface de la TST, et pour les deux modules intermédiaires à ces niveaux.

Les résultats obtenus par ce modèle sont ceux attendus, ce qui est un point très positif. Cependant, ce modèle se base sur un lexique (dans le sens de dictionnaire) restreint, il faudrait l'étendre afin d'avoir des résultats plus généraux et moins spécifiques. Un autre problème ce pose, car cette modélisation n'est que peu adaptée à la paraphrase syntaxique profonde : cette partie reste à améliorer. De plus, si les représentations de syntaxe profonde sont facilement compréhensibles, elles ne correspondent pas encore au texte écrit, un modélisation complémentaire des niveaux de représentation morphologiques profond et de surface peut être envisagée.

# Références


Abeillé, Anne (1993). *Les Nouvelles syntaxes, Grammaires d'unification et analyse du français*. Armand Colin. ISBN : 2-200-21096-5.

Berthoud, Françoise et al. (oct. 2019). *Sciences, Environnements et Sociétés*. Other. Inria. Archive ouverte HAL : hal-02340948.

Blom, Chris et al. (déc. 2011). "Implicit Arguments: Event Modification or Option Type Categories?" In : *18th Amsterdam Colloquium on Logic, Language and Meaning*. Sous la dir. de Maria Aloni et al. T. 7218. Lecture Notes in Computer Science. Amsterdam, Netherlands : Springer, p. 240-250. DOI : 10.1007/978-3-642-31482-7_25. Archive ouverte HAL : hal-00763102.

de Groote, Philippe (juill. 2001). "Towards Abstract Categorial Grammars". In : *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France : Association for Computational Linguistics, p. 252-259. DOI : 10.3115/1073012.1073045. Anthologie ACL : P01-1033.

Iordanskaja, Lidija, Richard Kittredge et Alain Polguère (1991). "Lexical Selection and Paraphrase in a Meaning-Text Generation Model". In : *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Sous la dir. de Cécile L. Paris, William R. Swartout et William C. Mann. Boston, MA : Springer US, p. 293-312. ISBN : 978-1-4757-5945-7. DOI : 10.1007/978-1-4757-5945-7_11. URL : https://doi.org/10.1007/978-1-4757-5945-7_11.

Lambrey, Florie et François Lareau (juin 2015). "Le traitement des collocations en génération de texte multilingue". In : *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles. Articles courts*. Caen, France : ATALA, p. 263-269. Anthologie ACL : 2015.jeptalnrecital-court.39.

Lareau, François et al. (mai 2018). "GenDR: A Generic Deep Realizer with Complex Lexicalization". In : *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan : European Language Resources Association (ELRA). Anthologie ACL : L18-1478.

Mel'čuk, I.A. et al. (2012). *Semantics: From Meaning to Text*. T. 1. Semantics: From Meaning to Text. John Benjamins Publishing Company. ISBN : 9789027205964.

— (2013). *Semantics: From Meaning to Text*. T. 2. Semantics: From Meaning to Text. John Benjamins Publishing Company. ISBN : 9789027206022.

— (2015). *Semantics: From Meaning to Text*. T. 3. Semantics: From Meaning to Text. John Benjamins Publishing Company. ISBN : 9789027259332.

Mel'Čuk, Igor et Alain Polguère (juill. 2021). "Les fonctions lexicales dernier cri". In : *La Théorie Sens-Texte. Concepts-clés et applications*. Sous la dir. de Sébastien Marengo. Dixit Grammatica. L'Harmattan, p. 75-155. Archive ouverte HAL : hal-03311348.

Milićević, Jasmina (2006). "A short guide to the Meaning-Text linguistic Theory". In : *Journal of Koralex* 8, p. 187-233.

Moortgat, Michael (1997). "Chapter 2 - Categorial Type Logics". In : *Handbook of Logic and Language*. Sous la dir. de Johan van Benthem et Alice ter Meulen. Elsevier, p. 93-177. DOI : 10.1016/B978-044481714-3/50005-9.

Morill, Glyn V. (1994). *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Publisher. ISBN : 0-7923-3095-1. DOI : 10.1007/978-94-011-1042-6.

Nasr, Alexis (1996). "Un modèle de reformulation automatique fondé sur la théorie sens-texte : application aux langues contrôlées". Thèse de doctorat d'informatique dirigée par Danlos, Laurence. Thèse de doct. Université Paris 7 - UFR d'Informatique, 277 p. URL : http://www.theses.fr/1996PA077102.

Pogodalla, Sylvain (2017a). "A syntax-semantics interface for Tree-Adjoining Grammars through Abstract Categorial Grammars". In : *Journal of Language Modelling* 5.3, p. 527-605. DOI : 10.15398/jlm.v5i3.193. Archive ouverte HAL : hal-01242154.

— (sept. 2017b). "Abstract Categorial Grammars as a Model of the Syntax-Semantics Interface for TAG". In : *FSMNLP 2017 and TAG+13 conference*. Umeå, Sweden. Archive ouverte HAL : hal-01583962.

Polguère, Alain (2003). *Lexicologie et sémantique lexicale. Notions fondamentales*. Paramètres. Les Presses de l'Université de Montréal.

Wanner, Leo et al. (2010). "MARQUIS: Generation of User-Tailored Multilingual Air Quality Bulletins". In : *Applied Artificial Intelligence* 24.10, p. 914-952. DOI : 10.1080/08839514.2010.529258. eprint :

https://doi.org/10.1080/08839514.2010.529258. URL : https://doi.org/10.1080/08839514.2010.529258.

# Index


**ACG** Grammaire catégorielle abstraite . . . 1
**ACGtk** ACG toolkit, Outil de développement catégoriel abstrait . . . 1

*Collocation* . . . 3
*Complexité d'une ACG* . . . 7

**DEC** Dictionnaire explicatif combinatoire . . . 13
**DMorph-ProsS** Structure morphologique profonde prosodique . . . 47
**DMorphR** Représentation morphologique profonde . . . 47
**DMorphS** Structure morphologique profonde . . . 47
**DPhon-ProsS** Structure phonologique profonde prosodique . . . 47
**DPhonR** Représentation phonologique profonde . . . 47
**DPhonS** Structure phonologique profonde . . . 47
**DSynt-AnaphS** Structure syntaxique profonde anaphorique . . . 10
**DSynt-CommS** Structure syntaxique profonde communicative . . . 10
**DSynt-ProsS** Structure syntaxique profonde prosodique . . . 10
**DSyntR** Représentation syntaxique profonde . . . 10
**DSyntS** Structure syntaxique profonde . . . 10

**FL** Fonction lexicale . . . 4
*Fonction lexicale* . . . 4

*Grammaire catégorielle abstraite* . . . 6

*Langage abstrait* . . . 6
*Langage objet* . . . 6
*Lexème* . . . 10
*Lexique* . . . 6

*Morphologie* . . . 49
**MST** Modèle sens-texte . . . 8

*Ordre d'une ACG* . . . 7

*Paraphrase* . . . 9
**PFE** Projet de fin d'études . . . 1
*Phonétique* . . . 47
*Phrasème* . . . 10

**RefS** Structure sémantique de référence . . . 9
*Relations syntaxiques profondes* . . . 10
*Rhème* . . . 45
**RhetS** Structure sémantique rhétorique . . . 9
**RLF** Réseau lexical du français . . . 13

*Sémantème* . . . 9
*Sémantique* . . . 2
**Sem-CommS** Structure sémantique communicative . . . 9
**SemR** Représentation sémantique . . . 9
**SemS** Structure sémantique . . . 9
*Sens* . . . 8
*Signature d'ordre supérieur* . . . 6
**SMorph-ProsS** Structure morphologique de surface prosodique . . . 47
**SMorphR** Représentation morphologique de surface . . . 47
**SMorphS** Structure morphologique de surface . . . 47
**SPhonR** Représentation phonologique de surface . . . 48

**SSynt-AnaphS** Structure syntaxique de surface anaphorique . . . 10
**SSynt-CommS** Structure syntaxique de surface communicative . . . 10
**SSynt-ProsS** Structure syntaxique de surface prosodique . . . 10
**SSyntR** Représentation syntaxique de surface . . . 10
**SSyntS** Structure syntaxique de surface . . . 10
*Syntaxe* . . . 2

**TAG** Grammaires d'arbres adjoints . . . 5
**TALN** Traitement automatique des langues naturelles . . . 2
*λ-termes linéaires* . . . 6
*Texte* . . . 8
*Thème* . . . 45
**TST** Théorie Sens-Texte . . . 1
*Types implicatifs linéaires* . . . 6

**UL** Unité lexicale . . . 2

# A Compléments sur la TST

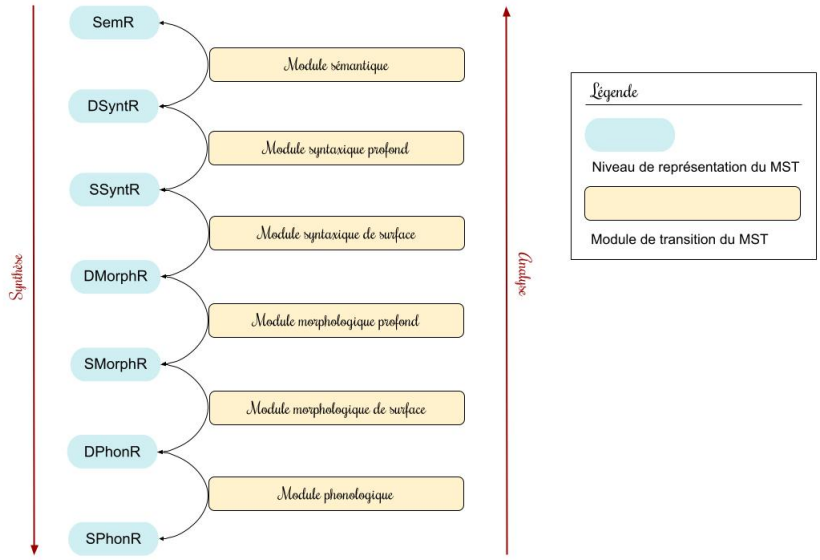



FIGURE 25 – Schéma complet d'un modèle de la TST

Chaque niveau de représentation possède plusieurs composantes, dont une principale. J'ai décrit (cf. partie 5) les composantes principales des sept représentations, et vais décrire ici les autres composantes de cinq des sept niveaux de représentation. En effet, je n'ai que très peu travaillé sur les deux niveaux phonologiques et n'ai aucun complément à ajouter pour ces niveaux là.

## A.1 SemR : la représentation sémantique

La représentation sémantique est composée de 4 structures. Sa structure principale est la structure sémantique **SemS**, et les trois autres sont les structures sémantique communicative **Sem-CommS**, rhétorique **RhetS**, et la structure de référence **RefS** (cf. figure 4 page 9).

$$\textbf{SemR} = \langle \text{ SemS, Sem-CommS, RhetS, RefS } \rangle$$

**La structure sémantique communicative Sem-CommS :** Elle spécifie l'organisation communicative de l'énoncé. Il y a huit dimensions communicatives. La première est la *thématique*, c'est-à-dire quelles parts de **SemS** est le thème et laquelle est le rhème, la seconde est la *donnée*, soit ce qui sera présenté comme information nouvelle et information connue ou présupposée. Les autres sont la *focalisation*, la *perspective*, l'*emphase*, l'*assertivité*, l'*unitarité*, et la *locutionalité*.
L'un des nœuds de **SemS** est marqué comme prédominant dans la phrase. Ce nœud $n_d$ est tel que tous les autres nœuds du graphe **SemS** sont dépendants communicativement de lui. C'est-à-dire que pour chaque nœud $n$ du graphe, il existe une suite d'arcs partant du nœud dominant $n_d$ et allant jusqu'à $n$ (Mel'čuk et al. 2012).

**Remarque A.1.1** *Ce nœud prédominant servira pour le choix de la racine pour l'arbre de la structure syntaxique profonde* ***DSyntS*** *(Mel'čuk et al. 2013).*

**Définition A.1.2** *Le* ***thème*** *d'une* ***SemS*** *est ce dont on parle, le sujet principal. Le* ***rhème*** *d'une* ***SemS*** *est ce qui est communiqué à propos du thème.*

**Exemple A.1.3** *Considérons les deux phrases suivantes :*
1. *Marie mange le gateau que Jean a fait.*
2. *Le gateau que Jean a fait est mangé par Marie.*

*Dans la phrase* 1.*, le thème est "Marie", et le rhème est "le gateau que Jean a fait", alors que dans la phrase* 2.*, c'est l'inverse : le thème est "le gateau que Jean a fait" et le rhème est "Marie".*

**La structure rhétorique RhetS :** Elle spécifie les propriétés rhétoriques de **SemS** (Mel'čuk et al. 2012). Elle décrit les intentions artistiques et les effets stylistiques voulus par le locuteur (i.e. neutre, officiel, ironique, sarcastique, poetique, etc.)

**Remarque A.1.4** *Plus **Sem-CommS** et **RhetS** sont spécifiées et détaillées, plus le nombre de paraphrases produites (ou productibles) sera petit. En effet, le thème et le rhème doivent être les mêmes d'une paraphrase à l'autre par exemple, ce qui contraint grandement le nombres de phrases considérées comme paraphrases valides pour un message spécifique (Mel'čuk et al. 2012). Par ailleurs, la phrase "Quelle personne sympathique ! [Ironique]", où [Ironique] est un marqueur de **RhetS**, ne veut pas dire la même chose que "Quelle personne sympathique ![Sérieux].*

**La structure de référence RefS :** Elle spécifie les liens entre les configurations sémantiques de **SemS** et les entités correspondantes dans le monde réel (Mel'čuk et al. 2012).

## A.2 DSyntR : la représentation syntaxique profonde

La représentation syntaxique profonde est composée de 4 structures. Sa structure principale est la structure syntaxique profonde **DSyntS**, et les trois autres sont la structure syntaxique profonde communicative **DSynt-CommS**, la structure syntaxique profonde prosodique **DSynt-ProsS**, et la structure syntaxique profonde anaphorique **DSynt-AnaphS** (cf.figure 6 page 10).

$$\textbf{DSyntR} = \langle\ \textbf{DSyntS}, \textbf{DSynt-CommS}, \textbf{DSynt-ProsS}, \textbf{DSynt-AnaphS}\ \rangle$$

**La structure syntaxique profonde communicative DSynt-CommS :** Elle est l'équivalent de **Sem-CommS**, i.e. elle attache les marqueurs des dimensions communicatives de **Sem-CommS** aux sous-arbres de **DSyntS**. Elle fournit l'information nécessaire au contrôle de la linéarisation et de la prosodisation (cf. partie A.3.5).

**La structure syntaxique profonde prosodique DSynt-ProsS :** Elle est constituée d'un ensemble de marqueurs qui indiquent la prose liée à l'énoncé, ou la **DSyntS** (déclaratif, neutre, ignorant, étonné, etc.). Ces marqueurs sont attachés aux sous-arbres de **DSyntS**.

**La structure syntaxique profonde anaphorique DSynt-AnaphS :** Elle donne les liens de co-référentialité entre les nœuds de la **DSyntS**. Une paire de nœuds co-référentiels sera reliée par une flèche en pointillés. Elle fournit l'information nécessaire pour faire des ellipses et les opérations de pronominalisation.

## A.3 SSyntR : la représentation syntaxique de surface

La représentation syntaxique de surface est composée de 4 structures. Sa structure principale est la structure syntaxique de surface **SMorphS**, et les trois autres sont les structures syntaxique de surface communicative **SSynt-CommS**, syntaxique de surface prosodique **SSynt-ProsS**, et syntaxique de surface anaphorique **SSynt-AnaphS** (cf.figure 8 page 10).

$$\textbf{SSyntR} = \langle\ \textbf{SSyntS}, \textbf{SSynt-CommS}, \textbf{SSynt-ProsS}, \textbf{SSynt-AnaphS}\ \rangle$$

**La structure syntaxique de surface communicative DSynt-CommS :** Elle est similaire à **DSynt-CommS**.

**La structure syntaxique de surface prosodique DSynt-ProsS :** Elle est similaire à **DSynt-ProsS**.

**La structure syntaxique de surface anaphorique DSynt-AnaphS :** Elle est similaire à **DSynt-AnaphS**.

### A.3.1 DMorphR : la représentation morphologique profonde

La représentation morphologique profonde est composée de 2 structures. Sa structure principale est la structure morphologique profonde **DMorphS** , et la deuxième structure est la structure morphologique profonde prosodique **DMorph-ProsS** (cf.figure 26)

**DMorphR** = ⟨ **DMorphS**, **DMorph-ProsS** ⟩

FIGURE 26 – Composantes de la **DMorphR**

**La structure morphologique profonde DMorphS :** Elle est représentée par une séquence linéaire de lexèmes décorés par leurs traits morphologiques ordonnés linéairement (Milićević 2006) (cf. figure 27).

$\text{I}_{NOM}$ + $\text{WILL}_{PRES}$ + $\text{DRAW}_{INF}$ + $\text{MARY}_{SG,POSS}$ + $\text{ATTENTION}_{SG}$ + TO + HER + $\text{FOOT}_{PL}$

FIGURE 27 – Exemple de structure morphologique profonde (Mel'čuk et al. 2012, p110) pour la phrase *I will draw Mary's attention to her feet.*

**La structure morphologique profonde prosodique DMorph-ProsS :** Elle indique les pauses, intonations et autres marques prosodiques, en venant annoter la **DMorphS**.

### A.3.2 SMorphR : la représentation morphologique de surface

La représentation morphologique de surface est composée de 2 structures. Sa structure principale est la structure morphologique de surface **SMorphS** , et la deuxième structure est la structure morphologique de surface prosodique **SMorph-ProsS** (cf.figure 28).

**SMorphR** = ⟨ **SMorphS**, **SMorph-ProsS** ⟩

FIGURE 28 – Composantes de la **SMorphR**

**La structure morphologique de surface SMorphS :** Elle est également représentée par une chaîne de caractères, mais contrairement à la **DMorphS**, les mots-formes sont remplacés par les mots eux-mêmes, en accord avec leurs traits morphologiques indiqués dans la **DMorphS** (Milićević 2006) (cf. figure 29).

{I},{$NOM$} + {WILL},{$PRES$} + {DRAW},{$INF$} + {MARY},{$'S$} +
{ATTENTION},{$SG$} + {TO} + {HER} + {FOOT},{$\text{A}_{PL}$}

FIGURE 29 – Exemple de structure morphologique de surface (Mel'čuk et al. 2012, p110) pour la phrase « *I will draw Mary's attention to her feet* ».

**La structure morphologique de surface prosodique SMorph-ProsS** Elle est similaire à la **DMorph-ProsS**.

### A.3.3 DPhonR : la représentation phonologique profonde

La représentation phonologique profonde représente la transcription phonétique (cf. definition A.3.1) de la représentation morphologique de surface (Mel'čuk et al. 2012). Elle est constituée des deux structures phonologique profonde (qui est sa structure principale, cf. figure 31 page suivante) et phonologique profonde prosodique.

**Définition A.3.1 (Polguère 2003)** *La **phonétique** (d'un énoncé) désigne les éléments sonores (de cet énoncé).*

**DPhonR** = ⟨ **DPhonS**, **DPhon-ProsS** ⟩

FIGURE 30 – Composantes de la **DPhonR**

/aⁱ wɪl drɔ́ mǽrɪz əténšn tu hɜr fít/

FIGURE 31 – Exemple de structure phonologique profonde (Mel'čuk et al. 2012, p111) pour la phrase « *I will draw Mary's attention to her feet* ».

### A.3.4 SPhonR : la représentation phonologique de surface

La représentation phonologique de surface est le dernier niveau de représentation (Mel'čuk et al. 2012). Je n'ai pas travaillé dessus.

### A.3.5 Le module syntaxique de surface : SSyntR ↔ DMorphR

Ce module permet la transition de la représentation syntaxique de surface à la représentation morphologique profonde (Milićević 2006). Je n'ai pas travaillé sur ce module, mais il permet de comprendre comment a lieu la transformation de la représentation de syntaxe de surface à la représentation morphologique profonde, beaucoup plus intuitive. Il est constitué de trois types de règles principalement :

— les règles de linéarisation : elles construisent une chaîne de caractères avec des mots ordonnés linéairement, à partir de la **SSyntR**.
— les règles de morphologisation : elles calculent les mots-formes pour chaque lexème, selon ses traits morphologiques.
— les règles de prosodisation : elles calculent les prosodies (les pauses, respirations, etc.) selon ce que la **SSyntR** indique.

# B Compléments sur les FLs

Les FLs sont organisées en deux grands groupes, les FLs standards et les FLs non standards. Cependant, nous nous intéressons aux FLs standards uniquement. Elles sont divisées en deux sous-groupes : les FLs syntagmatique et les FLs paradigmatiques.

**Les FLs syntagmatiques (Mel'Čuk et Polguère 2021) :** Elles traitent les collocations. Deux unités lexicales formant un phrasème (cf. definition 5.2.3 page 10) sont liées par une relation de collocation si la première est choisie indépendamment de la deuxième pour son sens, et la deuxième est choisie en fonction de la première pour exprimer une modification du sens de la première UL (cf. exemple B.0.1). On retrouve par exemple dans cette catégorie de FLs la fonctions d'intensité (`magn`).

**Exemple B.0.1** *Dans l'expression* « avoir une peur bleue »*, la première UL serait* PEUR *et la seconde* BLEUE*. En effet, la collocation* « peur bleue » *exprime une grande peur, donc* PEUR *est choisie pour son sens. Cependant, une peur n'est pas colorée littéralement parlant ;* BLEUE *vient donc compléter ou modifier le sens* '*peur*' *de* PEUR *dans cette collocation.*

**Les FLs paradigmatiques (Mel'Čuk et Polguère 2021) :** Elles expriment les relations de dérivation sémantique. Deux unités lexicales sont en relation de dérivation sémantique si il y a une d'une part différence entre leurs sens - on parle alors d'écart sémantique - et si d'autre part leur écart de sens se retrouve fréquemment dans la langue (cf. exemple B.0.3). L'un des cas particuliers de ces relations sont les relations de dérivation morphologique, où cette relation de dérivation sémantique est exprimée par des moyens morphologiques (cf. définition B.0.2). On retrouve notamment dans ces fonctions les FLs de synonymie (`syn`) ou de nom d'agent ($\mathtt{S}_1$).

**Définition B.0.2 (Polguère 2003)** *La* ***morphologie*** *(d'un énoncé) désigne la structure des mots (de cet énoncé). On retrouve par example les suffixes, préfixes, pluriels ou singuliers, temps et personnes de conjugaison.*

**Exemple B.0.3** $S_1$ *correspond souvent à une relation de dérivation morphologique.*
*En effet,* $S_1$*(manger)=mang****eur****,* $S_1$*(coiffer)=coiff****eur****, etc., où le suffixe -eur est ajouté dans la plupart des cas au radical.*

# C Compléments sur les ACGs

Je développe dans cette annexe un exemple d'ACG simple afin d'illustrer les notions introduites dans la section 4.1.

## C.1 Exemple d'ACG simple

**Signature $\Sigma_{string}$ :** Cette signature représente les chaînes de caractères, elle est constituée des éléments suivants :
- $o$, le type atomique,
- $\sigma = o \multimap o$, le type string,
- les constantes $John$, $Mary$, $assassinate$ de type $\sigma$,
- l'opérateur infixe pour la concaténation $+ = \lambda fg.\ \lambda z.\ f(g\ z)$ de type $\sigma \multimap \sigma \multimap \sigma$,

**Exemple C.1.1** *La phrase* « John assassinate Mary » *sera représentée par* $John + assassinate + Mary$.

**Signature $\Sigma_{tree}$ :** Cette signature représente des arbres, elle est constituée des éléments suivants :
- le type atomique $T$,
- les types $S = NP \multimap NP \multimap T$ et $NP = T$,
- les constantes $JOHN$, $MARY$ de type $NP$,
- la constante $ASSASSINATE$ de type $S$.

**Exemple C.1.2** *L'arbre illustré en figure 32 sera représenté par* $ASSASSINATE\ JOHN\ MARY$.

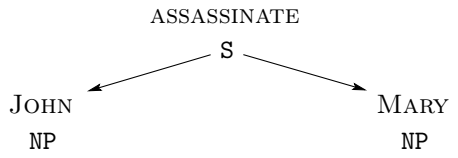



FIGURE 32 – Illustration de « *John assassinate Mary* » dans $\Lambda(\Sigma_{tree})$

**Lexique $\mathcal{L}_{tree\text{-}to\text{-}string}$ :** Ce lexique va de $\Sigma_{tree}$ dans $\Sigma_{string}$, et associe les éléments de $\Sigma_{tree}$ à ceux de $\Sigma_{string}$ de la manière suivante :
- $T := \sigma$
- $NP := \sigma$
- $S := \sigma$
- $JOHN := John$
- $MARY := Mary$
- $ASSASSINATE := \lambda\ x\ y.\ x + assassinate + y$

## C.2 Axiomes et règles d'inférences et exemple d'application

Les axiomes et règles d'inférence pour le $\lambda$-calcul sont les suivants (de Groote 2001) :

$$\frac{}{\vdash_\sigma c : \tau(c)}\ \text{axiome constante} \qquad \frac{\Gamma, x : \alpha \vdash_\sigma t : \beta}{\Gamma \vdash_\sigma \lambda x.t : \alpha \multimap \beta}\ \text{abstraction}$$

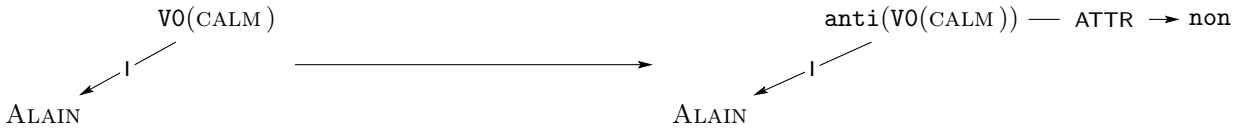
$$\frac{}{x : \alpha \vdash_\sigma x : \alpha}\ \text{axiome variable} \qquad \frac{\Gamma \vdash_\sigma t : \alpha \multimap \beta \quad \Delta \vdash_\sigma u : \alpha}{\Gamma, \Delta \vdash_\sigma (tu) : \beta}\ \text{application}$$

# D Exemple de paraphrase syntaxique pour la phrase « *Alain is calm* »

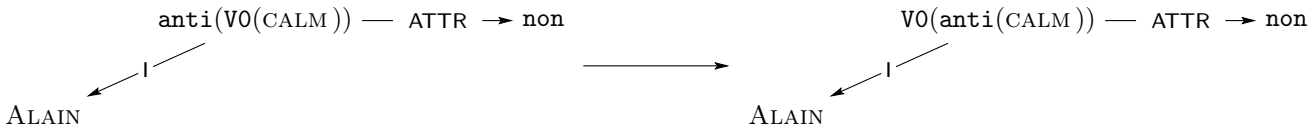
Voici les différentes étapes ayant lieu lors de la paraphrase syntaxique de « *Alain is calm* » en figure 33, suivies de la règle utilisée pour effectuer la première étape de cette paraphrase, en figure 33f.



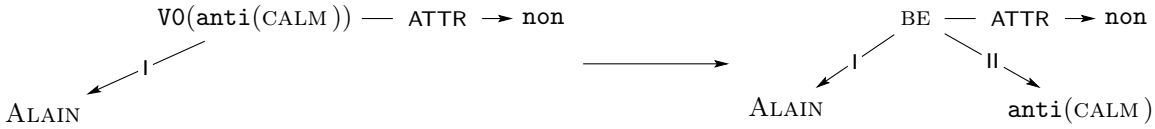

(a) Définition de V0 (cf. figure 33f)



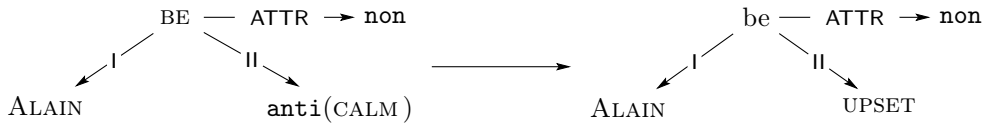

(b) Application de la règle lexicale 24 de paraphrase figure 6.4.1 page 24 et de la règle de restructuration support associée



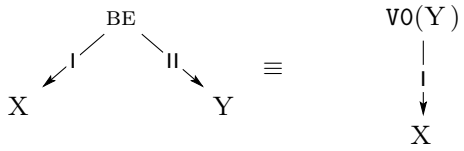

(c) Application de l'égalité Anti_(V0(L) = V0(Anti_(L)) (Mel'čuk et al. 2013)




(d) Application de la règle lexicale 24 de paraphrase figure 6.4.1 page 24 et de la règle de restructuration support associée




(e) Application de l'égalité anti(CALM) = UPSET (Mel'čuk et al. 2013)



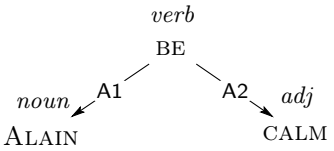

(f) Définition de V0

FIGURE 33 – Étapes de la paraphrase syntaxique pour la phrase « *Alain is calm.* »

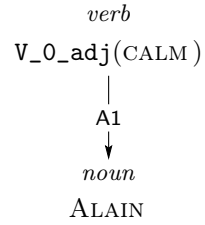
# E Paraphrase syntaxique et arbres de paraphrase générés

J'ai implémenté les trois règles présentées en annexe D (la définition de $\mathtt{V}_0$, la règle lexicale 24 et l'égalité de la figure 33e page précédente), et ai appliqué l'étape de paraphrase syntaxique sur l'arbre représentant « *Alain is calm.* ». Les arbres suivants ont été obtenus, en agissant comme je voudrais que le contrôleur agisse : à chaque étape, je ne traite donc que les nouveaux arbres obtenus, ceux déjà existant précédemment n'apporteront pas de nouvelle information.

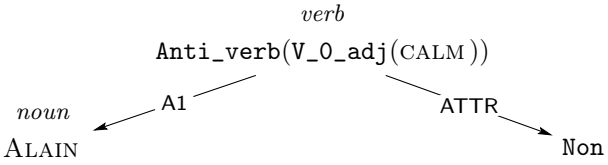
**Étape 1**



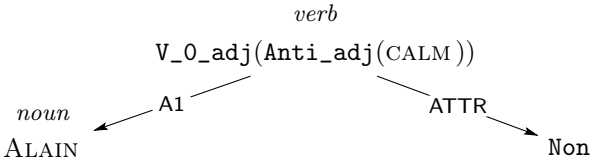

**Étape 2**



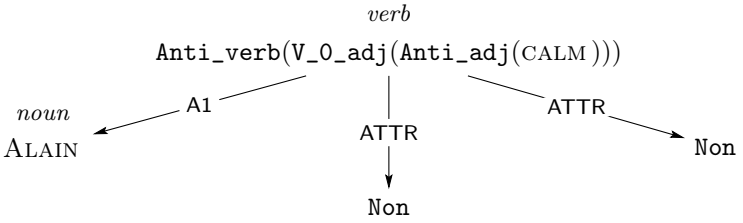

**Étape 3**



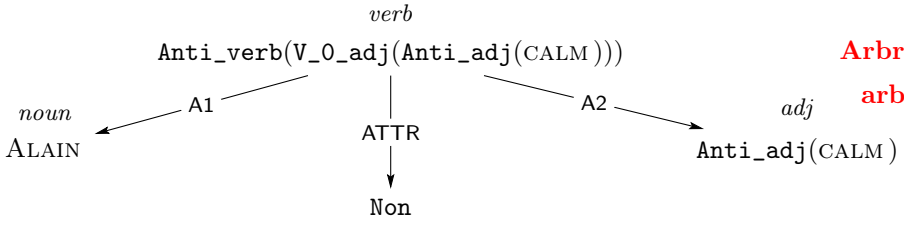

**Étape 4**




**Étape 5**




**Arbre incorrect**
**arrêt de la génération**




**Arbre correct - ne donne que des arbres obtenus précédemment lors de l'étape suivante de paraphrase**
**arrêt de la génération**

# F Exemple d'un script de test : « *John assasinate Mary.* »

```
sem parse ... : G;
dsyntRel realize ... : G;

lexfl parse ... : t;
reducefl realize ... : t;
reducefl parse ... : t;
lexfl realize ... : t;

compRel parse ... : T;
compEq realize ... : T;
compEq parse ... : T;
compRel realize ... : T;

lexfl parse ... : t;
reducefl realize ... : t;
reducefl parse ... : t;
lexfl realize ... : t;

lexfl parse ... : t;
reducefl realize ... : t;

ssynt parse ... : S/NP/AdvP/AdjP;
stree realize ... : S/NP/AdvP/AdjP;
```

FIGURE 34 – Ordre des commandes pour la transduction de $\Sigma_{semantic}$ à $\Sigma_{deep\text{-}rel}$, pour l'apparition des FLs, pour la paraphrase syntaxique profonde, et enfin la réalisation des FLs

```
# Phrase correcte
# John assassinates Mary

sem parse Lambda e0.
    Ex ex. Ex ey. Ex ez.
    (((((((murder e0) & (R1 e0 ex)) & (R2 e0 ey)) & (R3 e0 ez))
    & (john ex)) & (mary ey)) & (Ex_v v. v ez)) &
    (Ex e1. Ex e2.
    ((((reason e1) & (R1 e1 ex)) & (R2 e1 e0)) & (R3 e1 e2)) & (politics e2))
    : G;

dsyntRel realize c_assassinate (EXP c_john) c_mary : G;

lexfl parse verb ASSASSINATE
    ((A1 * (noun JOHN void)) & (A2 * (noun MARY void))) : t;
reducefl realize verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;
reducefl parse verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;
lexfl realize verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;

compRel parse verb ASSASSINATE
    ((A1 * (noun JOHN void)) & (A2 * (noun MARY void))) : T;
compEq realize l_verb l_ASSASSINATE
    ((l_A1 * (l_noun l_JOHN l_void)) & (l_A2 * (l_noun l_MARY l_void))) : T;
compEq parse e_verb e_ASSASSINATE
    ((e_A1 * (e_noun e_JOHN e_void)) & (e_A2 * (e_noun e_MARY e_void))) : T;
compRel realize l_verb l_ASSASSINATE
    ((l_A1 * (l_noun l_JOHN l_void)) & (l_A2 * (l_noun l_MARY l_void))) : T;

lexfl parse verb ASSASSINATE
    ((A1 * (noun JOHN void)) & (A2 * (noun MARY void))) : t;
reducefl realize verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;
reducefl parse verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;
lexfl realize verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;

lexfl parse verb ASSASSINATE
    ((A1 * (noun JOHN void)) & (A2 * (noun MARY void))) : t;
reducefl realize verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : t;

ssynt parse verb ASSASSINATE
    ((a1 * (noun JOHN void)) & (a2 * (noun MARY void))) : S;
stree realize ASSASSINATE12 JOHN0 MARY0 : S;

# verb ASSASSINATE ((Subj * (noun JOHN void)) & (DirO * (noun MARY void))) : T
```

FIGURE 35 – Script de test pour la phrase « *John assassinate Mary.* »

# G Diagramme de Gantt

Le diagramme de Gantt montrant mon avancement réel par rapport à mon avancement prévisionnel se trouve en page suivante.

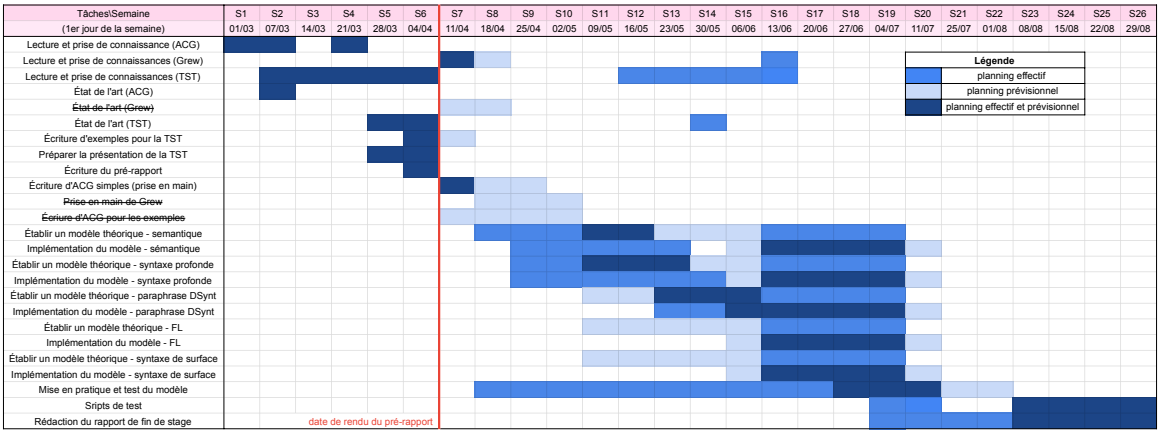



FIGURE 36 – Diagramme de Gantt

# H Extraits de code issus de ma modélisation

```
signature semantic =
    n : type;
    t : type;

    binder Ex : (n => t) -> t;
    binder Ex_v : ((n => t) => t) -> t;
    infix & : t -> t -> t;

    alain : n => t;
    assassinate : n => t;
    daughter_in_law : n => t;
    john : n => t;
    mary : n => t;
    murder : n => t;
    politics : n => t;
    reason : n => t;
    son : n => t;
    wife : n => t;

    R0 : n -> n -> t;
    R1 : n -> n -> t;
    R2 : n -> n -> t;
    R3 : n -> n -> t;
end
```

FIGURE 37 – Extrait de la signature $\Sigma_{semantic}$

```
lexicon sem(deep_syntactic):semantic=
    G := n => t;
    G' := n => t;
    MOD := t;

    IMP := Lambda e0. (Ex_v v. (v e0));
    EXP := lambda x. Lambda e0. x e0;

    I_MOD := true;

    c_for_pol_reas := lambda pred. lambda x. Lambda e0. Ex e1. Ex e3.
            (reason e1) & (R2 e1 e0) & (R3 e1 e3)
            & (pred true (Lambda e2. (R1 e1 e2) & (x e2)) e0) & (politics e3);
    c_mary := Lambda e0. (mary e0);
    c_murder := lambda A. lambda x y. Lambda e0. (Ex ex. Ex ey. Ex ez.
            (murder e0) & (R1 e0 ex) & (R2 e0 ey) & A
            & (x ex) & (y ey));
    c_john := Lambda e0. (john e0);
    c_son :=  lambda x y. Lambda e0. (Ex ex. Ex ey.
            (son e0) & (R1 e0 ex) & (R2 e0 ey) & (x ex) & (y ey));
    c_wife := lambda x y. Lambda e0. (Ex ex. Ex ey.
            (wife e0) & (R1 e0 ex) & (R2 e0 ey) & (x ex) & (y ey));
end
```

FIGURE 38 – Extrait du lexique $\mathcal{L}_{sem}$

```
signature deep_syntactic =
    G, G' : type;
    MOD : type;

    IMP : G';
    EXP : G -> G';

    I_MOD : MOD;

    c_alain : G;
    c_daughter_in_law_0 : G' -> G -> G;
    c_daughter_in_law_1 : G' -> G -> G;
    c_for_pol_reas : (MOD -> G' -> G) -> G' -> G;
    c_john : G;
    c_mary : G;
    c_murder : MOD -> G' -> G -> G;
    c_son : G' -> G -> G;
    c_wife : G' -> G -> G;
end
```

FIGURE 39 – Extrait de la signature $\Sigma_{deep\text{-}syntactic}$

```
signature deep_rel =
    N, V, Adj, Adv : type;
    T, Tb : type;
    rel : type;

    infix & : Tb -> Tb -> Tb;
    infix * : rel -> T -> Tb;
    verb : V -> Tb -> T;
    noun : N -> Tb -> T;
    adj : Adj -> Tb -> T;
    adv : Adv -> Tb -> T;

    A1 : rel;
    A2 : rel;
    A3 : rel;
    A4 : rel;
    ATTR : rel;
    COORD : rel;

    void : Tb;
    nil : T;

    V0_adj : Adj -> V;
    Anti_adj : Adj -> Adj;
    Anti_verb : V -> V;
    Non : T;

    ALAIN : N;
    ASSASSINATE : V;
    BE : V;
    CALM : Adj;
    C_FOR_POL_REAS : Adv;
    JOHN : N;
    MARY : N;
    MURDER : V;
    POLITICAL : Adj;
    REASON : N;
    SON : N;
    UPSET : Adj;
    WIFE : N;
end
```

FIGURE 40 – Extrait de code de la signature $\Sigma_{deep\text{-}rel}$

```
lexicon dsyntRel(deep_syntactic):deep_rel =
    G := T;
    G' := T;
    MOD := T;

    IMP := nil;
    EXP := lambda LEX. LEX;
    I_MOD := nil;

    c_alain := noun ALAIN void;
    c_assassinate := lambda X Y.  verb ASSASSINATE ((A1 * X) & (A2 * Y));
    c_be := lambda X Y. verb BE ((A1 * X) & (A2 * Y));
    c_calm := adj CALM void;
    c_for_pol_reas := lambda LEX. lambda X.
        LEX (noun REASON (ATTR * (adj POLITICAL void))) X;
    c_john := noun JOHN void;
    c_mary := noun MARY void;
    c_murder := lambda A. lambda X Y.
        verb MURDER ((A1 * X) & (A2 * Y) & (A3 * A));
    c_son := lambda X Y. noun SON ((A1 * X) & (A2 * Y));
    c_upset := adj UPSET void;
    c_wife := lambda X Y. noun WIFE ((A1 * X) & (A2 * Y));
end
```

FIGURE 41 – Extrait de code du lexique $\mathcal{L}_{dsyntRel}$

```
lexicon lexfl(fl):deep_rel =
    CALM1 := CALM;
    CALM2 := Anti_adj UPSET;
    RESTLESS := Anti_adj CALM;
    UPSET1 := UPSET;
    UPSET2 := Anti_adj CALM;
end
```

FIGURE 42 – Extrait de code du lexique $\mathcal{L}_{lexfl}$

```
lexicon reducefl(fl):fin_fl =
    CALM1 := CALM;
    CALM2 := CALM;
    RESTLESS := RESTLESS;
    UPSET1 := UPSET;
    UPSET2 := UPSET;
end
```

FIGURE 43 – Extrait de code du lexique $\mathcal{L}_{reducefl}$

```
signature surface_syntactic =
    S, NP, AdjP, AdvP : type;

    ALAIN0 : NP;
    BE12 : NP -> AdjP -> S;
    BE12N : NP -> AdjP -> S;
    CALM0 : AdjP;
    DAUGHTER_IN_LAW12 : NP -> NP -> NP;
    DAUGHTER_IN_LAW_2 : NP -> NP;
    JOHN0 : NP;
    MARY0 : NP;
    UPSET0 : AdjP;
    WIFE12 : NP -> NP -> NP;
    WIFE_2 : NP -> NP;
end
```

FIGURE 44 – Extraits de code de la signature $\Sigma_{surface\text{-}syntactic}$

```
lexicon ssynt(surface_syntactic):finfl =
    NP := t;
    S := t;
    AdjP := t;
    AdvP := t;

    ALAIN0 := noun0 ALAIN;
    BE12 := lambda X Y. verb12 BE X Y;
    BE12N := lambda X Y. verb12A BE X Y Non;
    CALM0 := adjec0 CALM;
    DAUGHTER_IN_LAW12 := lambda X Y. noun12 DAUGHTER_IN_LAW X Y;
    DAUGHTER_IN_LAW_2 := lambda Y. noun_2 DAUGHTER_IN_LAW Y;
    JOHN0 := noun0 JOHN;
    UPSET0 := adjec0 UPSET;
    WIFE12 := lambda X Y. noun12 WIFE X Y;
    WIFE_2 := lambda Y. noun_2 WIFE Y;
end
```

FIGURE 45 – Extraits de code du lexique $\mathcal{L}_{ssynt}$ Les patrons utilisés sont représentés par les constantes définies dans $\mathcal{L}_{fin\ fl}$ (cf. figure 47 page ci-contre), telles que *verb*12, qui prendra comme argument un actant syntaxique profond 1 et un actant syntaxique profond 2.

```
lexicon stree(surface_syntactic):surface_tree =
    NP := T;
    S := T;
    AdjP := T;
    AdvP := T;

    BE12 := lambda X Y. verb BE ((Subj * X) & (DirO * Y));
    BE12N := lambda X Y. verb BE ((Subj * X) & (DirO * Y)
        & (Adv_modif * (adv NOT void)));
    CALM0 := adj CALM void;
    DAUGHTER_IN_LAW12 := lambda X Y. noun DAUGHTER_IN_LAW ((Descr_relat * X)
        & (Attr * (prep OF (Prepos * Y))));
    DAUGHTER_IN_LAW_2 := lambda Y. noun DAUGHTER_IN_LAW (Poss * Y);
    JOHN0 := noun JOHN void;
    MARY0 := noun MARY void;
    UPSET0 := adj UPSET void;
    WIFE12 := lambda X Y. noun WIFE
        ((Descr_relat * X) & (Attr * (prep OF (Prepos * Y))));
    WIFE_2 := lambda Y. noun WIFE (Poss * Y);
end
```

FIGURE 46 – Extrait de code du lexique $\mathcal{L}_{stree}$.

```
signature finfl =
    verb1 = lambda lex. lambda X. verb lex (a1 * X)
        : v -> t -> t;
    verb1A = lambda lex. lambda X Y. verb lex ((a1 * X) & (attr * Y))
        : v -> t -> t -> t;
    verb1V = lambda lex. lambda X. verb lex ((a1 * X) & (attr * nil))
        : v -> t -> t;
    verb12 = lambda lex. lambda X Y. verb lex ((a1 * X) & (a2 * Y))
        : v -> t -> t -> t;
    verb1_ = lambda lex. lambda X. verb lex ((a1 * X) & (a2 * nil))
        : v -> t -> t;
    verb1_A = lambda lex. lambda X Y. verb lex ((a1 * X) & (a2 * nil) & (attr * Y))
        : v -> t -> t -> t;
end
```

FIGURE 47 – Extrait de $\Sigma_{finfl}$, où les patrons de structures d'arbres utilisés dans la représentation de syntaxe de surface sont utilisés