Développement de ressources lituaniennes pour un générateur automatique de texte multilingue

Ieva Dubinskaite

Sous la direction de François Lareau

Stage effectué à l'Observatoire de linguistique Sens – Texte
Université de Montréal
Département de linguistique et de traduction

Mémoire de master 2 Sciences du langage - 20 crédits

Parcours : Industries de la langue

Année universitaire 2016-2017

Remerciements

Je tiens tout d'abord à adresser mes remerciements les plus sincères à mon directeur de mémoire, François Lareau, pour sa grande disponibilité, ses encouragements ainsi que ses conseils lors de la rédaction de ce mémoire.

Je souhaiterais également remercier Agnès Tutin pour l'opportunité de partir à Montréal. Je n'oublierai jamais cette expérience.

J'adresse aussi mes remerciements à Florie Lambrey, Maryam Nejat et Daphnée Azoulay.

Sommaire

Introduction

La génération automatique de texte est une sous-branche du traitement automatique des langues qui vise à produire des textes dans une langue particulière à partir de données de nature non langagière, comme des données numériques brutes, des représentations logiques ou d'autres connaissances enregistrées dans des bases de données. Certains générateurs de texte sont multilingues et produisent des énoncés dans plusieurs langues. La conception d'un générateur de texte multilingue générique qui soit facilement réutilisable d'une application à l'autre nécessite de multiples ressources lexicales et grammaticales pour chacune des langues concernées. Bien que les ressources pour la génération soient de plus en plus disponibles pour les langues richement dotées, comme l'anglais et le français, les langues peu dotées, comme le lituanien, n'en disposent souvent pas. Si on veut concevoir un générateur de texte pour de telles langues, une question préliminaire doit être résolue : comment les ressources linguistiques peuvent-elles être obtenues de la manière la plus rapide et efficace possible ?

Pour développer un système de génération pour une grande variété de langues, il faut adapter les modèles linguistiques pour chaque langue. Il existe deux stratégies de conception de modèles linguistiques : l'adaptation de grammaires et le partage de grammaires. La première approche vise à réutiliser les grammaires existantes pour créer de nouvelles grammaires (Alshawi & Pulman, 1992 ; Kim et al., 2003) tandis que la deuxième cherche à définir quels phénomènes linguistiques peuvent être partagés entre les langues. Afin de créer une telle grammaire, il faut la modéliser à partir des similarités présentes dans les grammaires des différentes langues à traiter (Avgustinova & Uszkoreit, 2000 ; Bateman et al., 2005 ; Lareau & Wanner, 2007 ; Santaholma, 2008). Contrairement à la première approche, qui est considérée comme « faiblement multilingue » car elle ne peut être appliquée que pour des langues typologiquement similaires, la seconde approche est plus flexible et est capable de traiter des langues dont la grammaire et radicalement différente. Dans le cadre de ce mémoire, nous nous intéressons à cette seconde approche.

Récemment, quelques travaux ont porté sur le partage de ressources linguistiques appliqué à la génération automatique de texte multilingue, notamment Bateman et al. (2005) et Lareau & Wanner (2007). Dans le cadre du projet MARQUIS (Multimodal Air Quality Information Service for General Public) (Wanner et al., 2007, 2010), huit langues étaient couvertes :

catalan, anglais, français, polonais, portugais, finnois, allemand et espagnol. Il s'est avéré que l'implémentation de dictionnaires riches permet de concevoir des règles grammaticales plus génériques qui peuvent être utilisées pour plusieurs langues de différentes familles.

Le premier objectif de ce mémoire est d'intégrer un module de génération du lituanien dans GÉCO, un générateur automatique de texte multilingue exploitant des ressources linguistiques riches fondées sur le principe du partage de grammaires (Lambrey & Lareau, 2015 ; Lambrey, 2017). Ce générateur est en cours de développement à l'Observatoire de linguistique Sens-Texte de l'Université de Montréal. Il est basé sur la grammaire de MARQUIS et vise à implémenter de manière exhaustive les collocations dans le cadre de la génération automatique de texte multilingue. Notre deuxième objectif est de générer des collocations en lituanien en testant ainsi la performance du système de GÉCO.

Dans la première partie de ce mémoire, nous donnons un aperçu général sur la génération automatique de texte monolingue et multilingue ainsi que sur la lexicalisation et les collocations. Dans cette partie, nous décrivons également le fonctionnement et la réalisation linguistique de GÉCO. La deuxième partie du mémoire est destinée à décrire les tâches effectuées afin d'introduire un module du lituanien dans GÉCO et de présenter le difficultés rencontrés et les résultats obtenus.

Partie 1

-

État de l’art

Chapitre 1. Génération automatique de texte

Dans ce chapitre, nous introduisons la génération automatique de texte (GAT). Après avoir donné un aperçu général de cette discipline, nous présenterons l'entrée et la sortie d'un système de GAT et décrirons « l'architecture de consensus » partagée par beaucoup de systèmes.

La génération automatique de texte est une sous-discipline de l'intelligence artificielle qui vise à produire des textes en langage naturel à partir de sources d'information sous-jacentes de nature non linguistique. Un système de GAT est capable de prendre en entrée des données numériques de différents types, comme des données de capteurs GPS, signaux physiologiques, tableaux numériques, et produire des rapports, bulletins, explications, messages ou autres types de textes qui sont « grammaticalement corrects, sémantiquement cohérents et pragmatiquement pertinents » (Danlos & Roussarie, 2000). La génération de texte peut être considérée comme l'opposé de la compréhension automatique puisqu'elle a pour objectif d'exprimer le sens au lieu de le comprendre.

Historiquement, le domaine de la génération automatique est beaucoup plus récent que celui de la compréhension automatique. De nombreux programmes d'analyse ont vu le jour dès les années 1950, tandis que les premiers programmes de génération datent de la fin des années 1960 dans le cadre de projets de traduction automatique. Les premiers travaux sur la GAT destinée à communiquer de l'information de nature non linguistique sont apparus dans les années 1970 : notamment les recherches de Goldman (1975) sur le choix des mots appropriés pour exprimer le contenu conceptuel abstrait, et de Davey (1979) sur la génération de structures textuelles appropriées et expressions référentielles dans les descriptions de jeux tic-tac-toe (Reiter & Dale, 2000).

Aujourd'hui, les techniques de GAT vont des systèmes de génération de phrases les plus simples aux systèmes les plus sophistiqués de dialogue homme-machine. La GAT est mise en œuvre dans plusieurs domaines d'application : systèmes de prévision météorologique (Goldberg et al., 1994), génération d'histoires de fiction (Callaway & Lester, 2002), résumés textuels de données médicales (Portet et al., 2009), récits à partir de données de GPS (Baez et al., 2015), systèmes de tutoriels (Haller & Di Eugenio, 2003), génération de rapports statistiques (Iordanskaja et al., 1992), etc.

À l'ère du *big data*, l'analyse des données n'est plus suffisante. Transmettre à l'humain ce qui l'intéresse dans cette « infobésité » est d'une importance cruciale, d'où la question de comment présenter l'information : l'utilisateur veut comprendre vite, donc le texte proposé doit être le plus clair possible pour lui, ce qui nécessite un travail d'adaptation à son profil.

1. Entrée et sortie

Afin de parvenir à transformer les données en un texte cohérent de qualité satisfaisante, un système de GAT doit avoir accès à plusieurs connaissances. Reiter & Dale (2000, p. 43) définissent de façon formelle l'entrée de système de GAT typique sous forme de tuple ⟨*k*, *c*, *u*, *d*⟩, où :

- *k* est la source de connaissances spécifique au domaine d'application, encodée dans une ou plusieurs bases de données ou bases de connaissances ;
- *c* est le but communicatif que le système doit atteindre ;
- *u* est le modèle du public cible pour lequel le texte doit être généré ;
- *d* est l'historique de discours de ce qui a été dit dans les textes (d'un domaine particulier) produits jusqu'à présent.

Cette formule peut également être exprimée ainsi : avec les données *k*, on veut dire *c* à *u*, en utilisant le style *d*.

La sortie d'un système de GAT est principalement un texte qui correspond aux exigences de la demande posée à l'entrée. Comme l'illustrent Reiter & Dale (2000, p. 81), cela correspond généralement à une chaîne de mots, qui forment des phrases, des paragraphes et des sections en fonction du domaine d'application. La longueur du texte varie considérablement, pouvant aller d'un unique mot à un document de plusieurs paragraphes. Le texte généré doit respecter les normes de la langue, répondre à un objectif communicatif et être adapté au destinataire.

2. « Architecture consensuelle »

Afin de réaliser la transformation de l'entrée en un texte cohérent en langue naturelle, le système de GAT doit être capable de répondre à deux questions majeures : « quoi dire ? » et « comment dire ? » (Danlos & Roussarie, 2000). La première pourrait être paraphrasée comme suit : étant donné un ensemble d'informations et de connaissances à disposition d'un système de GAT, quelles sont les informations pertinentes à communiquer ? La deuxième

interrogation demande comment ces informations peuvent être traduites en langue naturelle, ou, autrement dit, comment structurer le texte en phrases et choisir les mots appropriés pour exprimer le contenu sélectionné dans l'étape précédente.

Reprenant ces deux questions, Reiter & Dale (2000) proposent une répartition de l'architecture en trois parties. Celle-ci est considérée comme « consensuelle » car elle correspond à l'architecture de la majorité de systèmes opérationnels. Le modèle proposé par Reiter & Dale (2000) est divisé en trois modules : macroplanification (plan du document), microplanification (spécification du texte) et réalisation de surface. Chacun peut être spécifié en présentant leurs tâches, conceptuellement distinctes (cf. Figure 1)

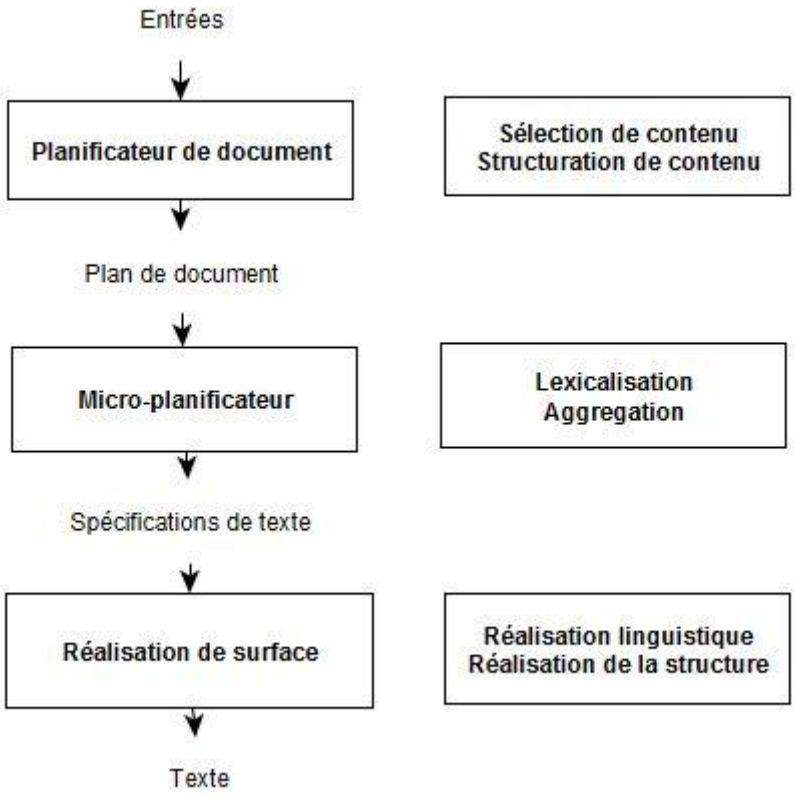


Figure 1 Architecture d'un système de GAT

Les lignes qui suivent expliquent plus en détail les tâches qui doivent être effectuées dans chacun des modules, en reprenant les points développés dans (Reiter & Dale 2000, pp. 49-81).

La **sélection de contenu** consiste à décider quelles informations doivent être communiquées dans le texte.

La **structuration de document** construit un ordre et une structure sur l'ensemble des messages à transmettre. L'information est extraite des données d'entrée, transformée en unités discursives (messages) et organisée à l'aide de relations rhétoriques.

La **lexicalisation** est le choix des mots et des structures syntaxiques qui doivent être utilisés pour exprimer le contenu sélectionné.

La **génération des expressions référentielles** consiste à sélectionner des mots ou des expressions pour identifier les entités du domaine.

L'**agrégation** consiste à agréger et ordonner les éléments de structures pour former des phrases et des paragraphes.

La **réalisation linguistique** génère des phrases grammaticalement correctes pour transmettre des messages.

La **réalisation de la structure** consiste à présenter le texte généré dans un format requis.

Dans cette architecture tripartite, l'enchainement des tâches forme le processus de génération de texte. C'est un processus linéaire ; autrement dit, la sortie du planificateur de document est l'entrée du micro-planificateur, dont la sortie est l'entrée de la réalisation de surface. Notons que le micro-planificateur ne peut pas influencer le comportement du planificateur de document, ni le réalisateur de surface celui du micro-planificateur ou du planificateur de document. Bien que cette architecture présente certaines limites, elle est utilisée dans de nombreux systèmes de GAT.

Regardons maintenant les spécificités de la génération automatique de texte multilingue (GATM).

Chapitre 2. Génération automatique de texte multilingue

Un générateur de texte multilingue produit des énoncés dans plusieurs langues. En présentant la GATM, (Lareau et al., 2011) cite divers types de textes : rapports statistiques gouvernementaux (Iordanskaja et al., 1992), contes de fées (Callaway & Lester, 2002), visites de musées (Callaway et al., 2005), terminologie médicale (Rassinoux et al., 2007), etc. La conception de tels systèmes requiert de multiples ressources lexicales et grammaticales. Afin de réduire le coût et la durée de leur développement, il est essentiel d'optimiser le partage des ressources linguistiques entre les langues à traiter. Dans les sections suivantes, nous regarderons quelles sont les spécificités de l'architecture typique d'un système de GATM et verrons comment les ressources linguistiques multilingues sont partagées selon les approches de l'adaptation de grammaires et du partage de grammaires dont nous avons parlé antérieurement dans l'introduction.

1. Architecture d'un système GATM

Dans le chapitre précédent, nous avons vu que la plupart des systèmes de GAT sont modulaires. Selon Bateman et al. (1999), un des avantages d'une telle architecture est qu'elle permet de séparer les composants indépendants de la langue de ceux qui en dépendent, permettant d'ajouter le multilinguisme avec beaucoup moins de travail qu'il n'en faudrait avec la construction de description linguistique pour chaque langue à partir de zéro. Marcu et al. (2000) mettent également l'accent sur l'importance de séparer les modules conceptuels et linguistiques. Une telle division du système permet une solution plus viable d'un point de vue économique car la plupart des algorithmes de génération peuvent être réutilisés. Lambrey (2016) cite Reiter & Dale (2000, p. 55) et Steinlin (2003), qui perçoivent la génération multilingue comme un module de génération unique sur lequel s'appuient plusieurs modules adaptés à une langue particulière. Selon ces auteurs, un texte généré dans plusieurs langues transmet le même contenu, seule la réalisation linguistique de ce contenu diffère d'une langue à l'autre.

2. Partage de ressources linguistiques

Le développement de la grammaire constitue une grande partie du cycle de développement d'un système multilingue à base de règles. Une façon de réduire l'effort est de partager les ressources linguistiques entre les langues. Le partage de la grammaire réduit la quantité de ressources nécessaires pour une langue donnée car les règles centrales ne sont

introduites qu'une seule fois. Cela conduit automatiquement à une cohérence entre les descriptions pour différentes langues, ce qui améliore la maintenabilité de la grammaire et diminue la taille du code (Santaholma, 2008). Comme nous avons déjà vu précédemment, il existe deux stratégies de partage de ressources linguistiques : adaptation de grammaires (*grammar porting*) et partage de grammaires (*grammar sharing*). La première vise à modifier une grammaire déjà existante pour couvrir une nouvelle langue tandis que dans la deuxième les règles sont directement partagées entre les langues plutôt que simplement recyclées comme dans l'adaptation de grammaires. Regardons maintenant de plus près les spécificités de ces deux stratégies.

Commençons par l'adaptation de grammaires. Santaholma (2008) définit cette stratégie comme une technique qui vise à modifier une grammaire existante afin de couvrir une nouvelle langue. Cette approche est appliquée par (Kim et al., 2003) dans le cadre du projet ParGram (Parallel Grammar Project). ParGram est un système d'analyse basé sur le parseur XLE et une plateforme de développement de grammaires. L'objectif du projet est de produire des grammaires à large couverture pour diverses langues. Comme indiquent (Butt et al. 2002), le projet ParGram tente également d'expérimenter l'universalité du formalisme LFG (Kaplan & Bresnan, 1982 ; Dalrymple, 2001) et d'examiner dans quelle mesure le parallélisme peut être maintenu à travers les langues. Basées sur LFG, les grammaires de ParGram couvrent plusieurs langues typologiquement différentes : l'anglais, le français, l'allemand, le japonais, le norvégien et l'urdu. Dans le cadre de ce projet, (Kim et al., 2003) étaient intéressés à l'adaptation de la grammaire du japonais vers le coréen. Les auteurs ont mis en évidence le fait que dans LFG, les langues typologiquement similaires possèdent des f- structures proches. En se basant sur cette similarité des grammaires, la grammaire du coréen a été construite en important la majorité des règles sans changements (par exemple, l'ordre des mots) et en retravaillant certains phénomènes (négation phrastique et double accusatif). La construction de la grammaire a duré seulement deux mois. La plupart des règles de structure de phrases sont restées les mêmes et de nombreux éléments lexicaux ont pu être importés semi-automatiquement en changeant le mot-clé de l'entrée japonaise par son équivalent coréen. Malgré l'efficacité évidente de cette approche, son application n'est possible qu'entre des langues typologiquement similaires.

Afin de partager les ressources linguistiques entre des langues typologiquement différentes, la stratégie qui s'avère plus efficace est le partage de grammaires. Les grammaires multilingues peuvent partager les ressources de différentes façons. Bateman et

al. (2005) proposent une méthode basée sur l'hypothèse que les langues partagent les fonctions communicatives, bien qu'elles puissent différer dans la réalisation de ces fonctions :

> *The strong notion of multilinguality arises from a functionalist perspective on grammar organization: Although form may vary across languages, we hypothesize that the underlying communicative functions are largely shared. In other words, we take paradigmatic choices to be more often congruent across languages than the syntagmatic ones.*

Cette approche repose sur les similarités et différences fonctionnelles entre les langues : en modélisant les différentes fonctions, le niveau de description linguistique qui transcende les langues est beaucoup plus efficace que leurs descriptions structurelles. La grammaire fonctionnelle systémique (Halliday, 1967) est le formalisme créé à partir de cette hypothèse. Il constitue la base théorique du générateur automatique de texte multilingue KPML (Bateman, 1997). La grammaire de KPML est encodée comme un réseau de systèmes orienté qui couvre plusieurs niveaux de représentation linguistique (syntaxe, morphologie, etc.). Chaque croisement expose un choix entre différents traits fonctionnels. Une telle architecture est représentée dans la Figure 2.

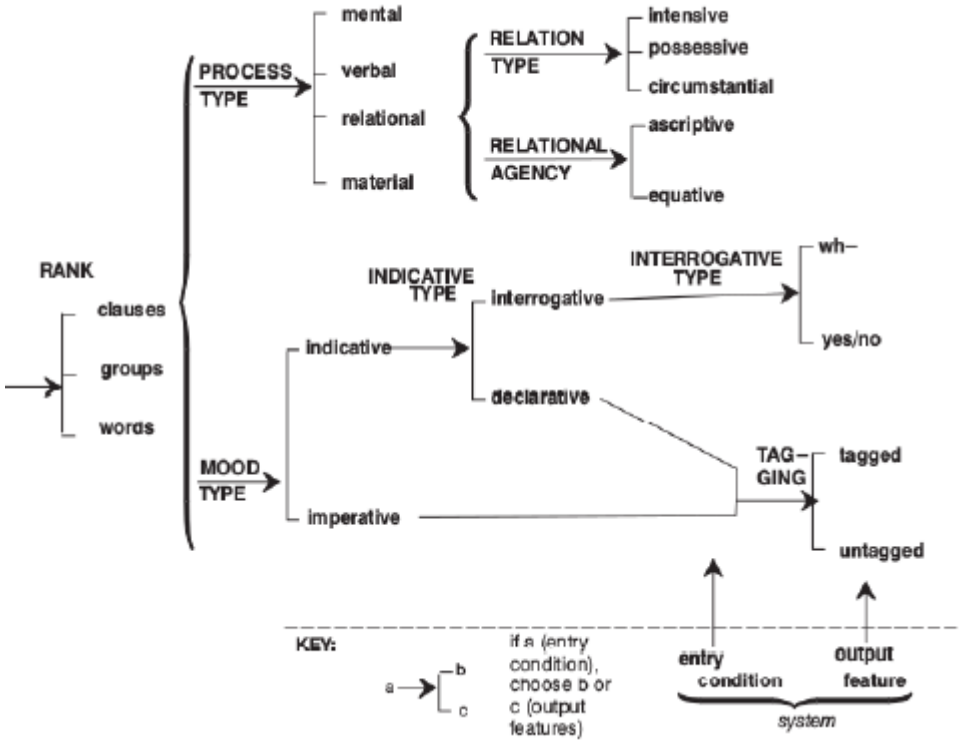


Figure 2 Réseau de systèmes de KPML (Bateman, 1997)

Dans le cadre du projet Agile, Bateman et al. (2005) ont illustré la mise en œuvre et les avantages de cette approche à l'égard de quatre langues couvertes par le générateur KPML : l'anglais, le bulgare, le tchèque et le russe. La Figure 3 représente les statistiques de réutilisation des réseaux de grammaire des langues couvertes : les nombres entre parenthèses (en dehors) désignent des calculs de caractéristiques partagées.

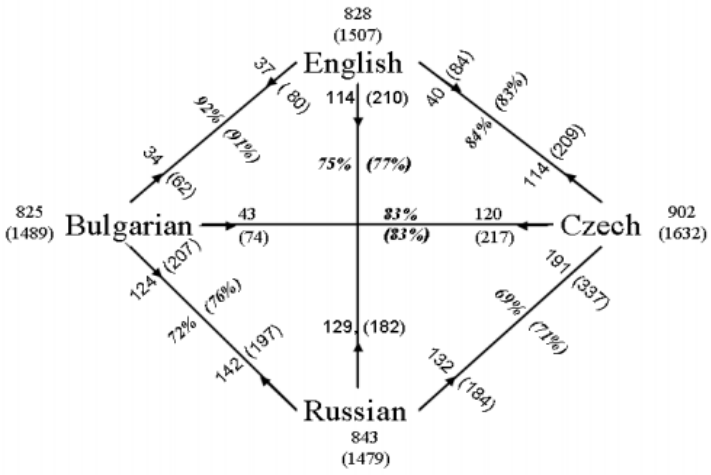


Figure 3 Statistiques de partage de ressources dans le cadre de projet Agile (Bateman et al., 2005)

Alors que le système de Bateman (2005) propose une méthodologie fonctionnaliste, Santaholma (2008) suggère une approche qui s'appuie sur la structure de la langue. Cette approche repose sur le partage de ressources grammaticales communes entre les langues. Dans le cadre de la grammaire Regulus, quatre langues typologiquement différentes sont couvertes : l'anglais, le finnois, le japonais et le grec. La grammaire comporte 80 règles, dont 43 sont partagées entre les langues du système. Les règles couvrent les phénomènes linguistiques principaux tels que la sous-catégorisation, la voix, le type de déterminant, etc. La grammaire multilingue de Regulus est modulaire et organisée hiérarchiquement. Les règles paramétrées sont stockées dans un module principal « indépendant de la langue » qui est le niveau le plus générique et partagé entre toutes les langues. Les niveaux inférieurs comprennent les modules spécifiques à une langue donnée. Les informations de cette structure modulaire sont héritées du haut vers le bas, du plus générique au plus spécifique. Les paramètres qui correspondent à chaque langue sont insérés à l'aide de macros qui correspondent à des généralisations dans les règles grammaticales multilingues, et particulièrement dans le lexique. Santaholma (2008) a démontré qu'une telle architecture permet la réutilisation de règles pour une nouvelle langue dans un domaine limité. Le

déploiement d'une nouvelle langue est rapide car il repose principalement sur des règles déjà existantes : l'implémentation du grec a été effectuée en seulement deux semaines environ.

Le principe de partage des ressources linguistiques s'appuie sur la conception de grammaire abstraite qui décrit les phénomènes linguistiques récurrents entre les langues. Nous venons de voir que dans le cadre de GATM, ces phénomènes sont réalisés différemment dans l'étape de lexicalisation selon chaque langue en question. Comme l'indique (Lambrey, 2016), la lexicalisation joue un grand rôle dans la gestion de plusieurs langues et elle peut servir également pour traiter les collocations. Dans le cadre du projet GÉCO, elle cherchait à modéliser les collocations à un niveau suffisamment abstrait où s'opère la lexicalisation. La lexicalisation et les collocations sont le sujet du prochain chapitre.

Chapitre 3. Lexicalisation et collocations

Dans ce chapitre nous nous intéressons au processus de lexicalisation et aux différents problèmes liés à celui-ci. Nous présentons également les collocations et les fonctions lexicales (FL) et décrivons quelques stratégies d'implémentation des FL qui ont été appliquées dans le contexte de GATM.

Comme défini dans (Reiter & Dale, 2000, p. 124), « the lexicalisation is the process of choosing words and syntactic structures to communicate the information in a document plan. This means mapping the messages in the document plan, which are expressed in terms of domain model, into words and other linguistic resources which make sense to the intended user of the document ». La lexicalisation n'est pas une tâche aussi aisée qu'elle puisse paraître : il ne suffit pas tout simplement d'introduire les unités lexicales au lieu des concepts. Comme le remarque Steinlin (2003 p. 21) « les concepts ne sont pas nécessairement dans une relation biunivoque avec les unités lexicales », et, « établir une relation d'un-à-un entre les concepts et les lexèmes reviendrait à laisser de côté la synonymie, une caractéristique pourtant importante des langues ». Il est important de souligner qu'une telle correspondance ne pourrait pas être appliquée dans la GATM, car « il est rare en effet de trouver entre deux lexies appartenant à deux langues un recouvrement exact de la dénotation conceptuelle » (*ibid.*).

Steinlin (2003) reprend les cinq configurations de correspondances entre concept et lexème de Stede (1996) :

1. Le choix lexical correspond à une relation un-à-un entre le concept et le lexème.
2. Le concept correspond à plusieurs lexèmes entretenant des relations de synonymie.
3. Le concept se lexicalise par le lexème correspondant ou un hyponyme.
4. Plusieurs concepts se lexicalisent par la même lexie (homonymie).
5. Le concept ne possède pas de correspondant lexical dans la langue.

L'interdépendance entre les choix lexicaux et les constructions syntaxiques est un autre problème qui se pose lors de la lexicalisation. Lambrey (2016) et Steinlin (2003) l'illustrent dans le choix de la partie du discours : le sens 'amour' peut se lexicaliser de trois

manières différentes : le nom AMOUR, le verbe AIMER et la construction ÊTRE AMOUREUX. Le choix d'une partie du discours aura une influence sur les positions que le lexème occupera dans la phrase.

En raison de la complexité de lexicalisation, Polguère (1998) propose de répartir le processus sur plusieurs étapes. Une telle méthodologie a été appliquée dans G-TAG (Steinlin, 2003), où la lexicalisation est repartie en deux niveaux et, comme nous le verrons plus tard, dans MARQUIS et GÉCO, qui comportent trois niveaux.

Dans ce mémoire, nous nous intéressons en particulier à la génération des collocations dans le contexte multilingue. Dans GÉCO, le processus de génération des collocations est mis en place au niveau de la lexicalisation. Avant de passer à la description de processus de lexicalisation de GÉCO, définissons tout d'abord la notion de collocation et regardons pourquoi le traitement des collocations est primordial dans la génération de texte.

1. Collocations et GAT

Comme nous venons de le voir, une des étapes de la réalisation linguistique d'un système de GAT est la lexicalisation, où des lexèmes spécifiques sont choisis pour exprimer le contenu du message. Nous avons également vu que pour certaines raisons, cela ne peut pas être tout simplement réalisé en reliant les concepts aux lexèmes qui leur correspondent. Dans le cas idéal, le concept RAIN pourrait être lié aux lexèmes *rain* en anglais, *lietus* en lituanien, *chuva* en portugais, etc. Pourtant, il arrive très souvent que les concepts doivent être lexicalisés d'une manière différente selon les lexèmes avec lesquels ils apparaissent. Considérons, par exemple, deux cas suivants en anglais :

- *Heavy rain*
- *Strong rain*

Quoiqu'on arrive à comprendre le sens de *strong rain*, un locuteur natif emploierait plutôt *heavy rain*. Bien que *rain* soit choisi librement, le lexème qui le modifie, *heavy*, ne l'est pas. *Strong* a à peu près la même signification que *heavy*, mais leur choix est lié au lexème qu'ils modifient, ici *rain*. Ces associations binaires de deux unités lexicales sont appelées collocations. Un système de génération de texte doit être capable de générer les collocations afin que les textes soient lisibles et simulent au mieux le langage naturel. Définissons maintenant les collocations de façon formelle et regardons comment elles sont modélisées via les fonctions lexicales.

2. Collocations et fonctions lexicales

On entend par collocations des associations récurrentes binaires de lexèmes, composées d'un prédicat et d'un argument où le choix du prédicat est déterminé par celui de l'argument, comme *peur bleue*, *brouillard dense*, *remercier chaleureusement*. Aussi appelées cooccurrences lexicales restreintes, elles sont très fréquentes dans la langue et tout locuteur natif est capable de les utiliser.

Dans le cadre de la Théorie Sens-Texte, les collocations sont modélisées via les fonctions lexicales (Žolkovskij & Mel'čuk, 1967 ; Mel'čuk, 1973, 1988, 1996, 1998; Kahane & Polguere, 2001 ; Kahane, 2003 ; Jousse, 2003, 2010). Depuis la création de la théorie, les fonctions lexicales ont été utilisées dans de nombreuses applications du TAL, notamment en traduction automatique et en génération automatique de texte monolingue et multilingue (Heid & Raab, 1989 ; Heylen, Maxwell, & Verhagen, 1994 ; Maxwell & Heylen, 1994, Steinlin, 2003; Lareau et al., 2011). Toutefois, elles n'ont jamais été couvertes de façon exhaustive dans un système de GAT, d'où la pertinence de GÉCO.

Dans cette section, nous donnons une vision globale de l'ensemble du système des fonctions lexicales : les fonctions lexicales standard et les fonctions lexicales non-standard. Nous nous basons essentiellement sur les travaux de Jousse (2003, 2010). Les exemples des FL sont repris des sources suivantes : (Jousse 2003, 2010), (Steinlin, 2003), (Lambrey, 2016).

3. Modélisation des fonctions lexicales

La Théorie Sens-Texte décrit les patrons de collocations au moyen de fonctions lexicales qui expriment les différents types de relations sémantico-lexicales entre deux lexèmes. Les fonctions lexicales ressemblent aux fonctions mathématiques, **f(x) = y**, et représentent une relation sémantique entre deux ensembles de lexies : **un mot-clé x (base),** auquel est associé un ensemble de valeurs **y (collocatifs)**.

Pour donner un exemple, prenons la fonction lexicale **Magn** (du lat. *magnus* 'grand') :

Magn(dormir) = *profondément, comme une souche, comme un loir, à poings fermés*

Magn(blessé) = *gravement, grièvement*

Comme nous pouvons le voir, la FL Magn exprime une relation sémantico-lexicale d'intensification. Autrement dit, si l'on applique Magn sur dormir, elle retournera la valeur profondément, etc.

L‘objectif des FL est de décrire les collocations de manière systématique et fonctionnelle. Jousse (2010) cite plusieurs types de classification des fonctions lexicales (Žolkovsky et Mel’čuk 1966, 1967, 1970; Mel’čuk, 1982; Mel’čuk et Žolkovsky 1984, 1988; Mel’čuk et al. 1984, 1988; Steele 1986; Steele et Meyer 1990; Grimes 1990; Alonso Ramos 1993; Mel’čuk et al. 1995; Alonso Ramos et Tutin, 1996; Fontenelle 1997). Dans les sections qui suivent, nous présentons succinctement le FL selon un premier axe de classification : fonctions standard et non standard. Cette classification est abordée plusieurs fois dans la littérature (Mel’čuk, 1996; Mel’čuk et al., 1999; Jousse, 2003, 2010; Polguère, 2007).

3.1. Fonctions lexicales standard

Les FL du type standard sont peu nombreuses. Il en existe environ une soixantaine. Citons les propriétés des FL standard présentées dans l’ILEC (p. 127-128) :

- En règle générale, f(L1) et f(L2) sont différents :$\mathbf{f}(L_1) \neq \mathbf{f}(L_2)$.
- La fonction **f** a un nombre élevé d'arguments (= de mots-clés). [En d'autres mots, le sens '**f**' est très abstrait et très général et s'applique à beaucoup d'autres sens.]
- La fonction **f** a un nombre élevé d'élément dans sa valeur (= d'expressions).

Les FL standard sont universelles : elles existent dans toutes les langues et suffisent à décrire l'ensemble des collocations de façon systématique et formelle.

Il existe deux classements des **FL standard simples** : celles qui expriment les relations paradigmatiques et celles qui expriment les relations syntagmatiques entre les lexies.

Les **FL paradigmatiques** expriment des relations de dérivation sémantique : l’équivalence, l’inclusion ou l’intersection. Voici quelques exemples de FL paradigmatiques :

- **Syn**(naïf) = *candide* (synonymie)
- **Conv**(vendre) = *acheter* (conversion)
- **Anti**(bien) = *mal* (antonymie)

Les **FL syntagmatiques** modélisent les relations syntaxiques entre unités lexicales qui apparaissent ensemble. Les FL syntagmatiques sont divisées en trois catégories : adjectivales, adverbiales et verbales.

Les **FL adjectivales** expriment une relation de type de modificateur :

- **Magn**(pleurer)= *comme une Madelaine* (intensificateur)
- **Ver**(explication)= *claire* (confirmateur)
- **Bon**(se porter)= *comme un charme* (laudatif)

Les **FL adverbiales** expriment, en général, le moyen, la localisation et la cause :

- **Instr**(téléphone) = *par* [~]
- **Loc$_{in/ad}$**(campagne) = *à la* [~]
- **Propt**(maladie) = *pour cause* [de ~]

Les **FL verbales** peuvent être distingués en quatre sous-ensembles selon le type du verbe :

Verbes supports, ou autrement dit, les verbes sémantiquement vides qui verbalisent des noms prédicatifs.

- **Oper$_1$**(attention) = *faire* ~ à N
- **Func$_0$**(pluie) = ART ~ *tomber*
- **Labor$_{12}$**(torture) = *soumettre* ~ à N

Verbes de réalisation, ou verbes sémantiquement pleins qui expriment les objectifs inhérents de la chose désignée par le mot clé :

- **Real$_1$**(promesse) = *tenir* ART ~
- **Fact$_0$**(rêve) = *se réaliser*
- **Labreal$_{12}$**(mémoire) = *conserver* ~ en N

Verbes phasiques (**Incep**, **Fin**, **Cont**) qui expriment les trois phases différentes d'un état ou d'un événement. Ils se combinent en général avec une FL verbale :

- **IncepOper$_1$**(amour) = *tomber*~ en
- **FinFunc$_0$** (blessure) = *se cicatriser*
- **ContOper$_1$** (sang-froid) = *garder*

Verbes causatifs (**Caus**, **Liqu**, **Perm**) qui expriment les trois types de causation d'un état ou d'un événement. Ils se combinent en général avec une FL verbale :

- **CausFunc$_1$**(forme) = donner à N ART ~

3.2. Combinaison des fonctions lexicales standard simples

Un des traits des FL est leur capacité combinatoire. Comme indiqué dans (Jousse, 2003), les FL standard simples peuvent se combiner de deux façons : on distingue les FL complexes et les configurations de fonctions.

Les **FL complexes** représentent les FL combinées de deux ou plusieurs FL. Voici la définition des FL complexes de l'ILEC (Mel'čuk et al., 1995: 148) reprise par (Jousse, 2003) :

> *Nous appelons **fonction lexicale complexe** un enchaînement de FL simples syntaxiquement liées, cet enchaînement ayant une valeur globale cumulative, qui exprime, de façon indécomposable, le sens de l'enchaînement entier.*

Voici quelques exemples de FL complexes :

- **IncepOper$_1$**(ami) = *devenir*
- **AntiBon**(choix) = *mauvais*

Les FL simples et les FL complexes peuvent être combinées et former des **configurations de FL**. Contrairement aux FL complexes, les constituants de configuration de FL ne sont pas syntaxiquement liés. Afin de donner la définition de configuration de FL, (Jousse, 2003) cite l'ILEC :

> *Nous appelons **configuration de fonctions lexicales** une suite de FL simples qui ne sont pas syntaxiquement liées entre elles, mais qui ont le même mot-clé, cette suite ayant une valeur globale cumulative qui exprime de façon indécomposable le sens de la suite entière.*

Voici quelques exemples de configurations de FL (Jousse, 2003) :

- **Bon**(joie) + **Magn**(joie) = paradisiaque : "une grande [**Magn**] joie qui est très bonne [**Bon**] "
- **Magn**(maladie) + **A$_1$** (maladie) = terrassé [par ART ~] : "qui a [**A$_1$**] une maladie grave [Magn]"
- **Magn**(orage) + **Caus$_2$Func$_0$**(orageII) = déchaîner [ART ~] : "X cause exister [**Caus$_2$Func$_0$**] un grand [**Magn**] orage au sujet de X"

3.3. Fonctions lexicales non standard

Les relations sémantiques dont le sens est très spécifique et qui ne peut pas être généralisé par des FL standard sont appelées non standard. Les **FL non standard** ne sont

pas universelles, elles ne sont pas « prévisibles et ne peuvent donc pas être dégagées et recensées de façon méthodique » (Mel'čuk et al., 1995 :150).

(Jousse, 2003) reprend un exemple d'une fonction lexicale non standard de l'ILEC (p. 150):

STEAK, nom, masc.

{à peine cuit} : saignant

{peu cuit} : bleu

{cuit} : à point

{S. garni de frites} : ~ frites

{S. garni de salade} : ~ salade

Dans ces exemples, la relation sémantico-lexicale entre les éléments est très spécifique et ne peut pas être appliqué à d'autres lexies. Les fonctions non-standard sont « surtout typiques pour des mots concrets, qui sont normalement très marqués culturellement ou techniquement : noms de nourriture (fromages, vins, etc.), de vêtements, de transactions financières, de procédures médicales, d'activités culturelles, politiques ou religieuses, d'armes, de parties du corps […]. Ces fonctions sont extrêmement nombreuses dans le secteur des langues spécialisées (technologie, enseignement, droit, …) ». En présentant les FL non standard, Lambrey (2016) conclut que « la distinction principale entre une fonction lexicale standard et une fonction lexicale non standard est quantitative. Cependant, c'est ce côté quantitatif qui permet de rendre une FL généralisable et systématique » (Lambrey, 2016, p.51). Dans notre mémoire, nous ne traitons pas les FL non standard, car elles ne peuvent pas être générées à l'aide de règles génériques et systématiques.

4. Utilisation des fonctions lexicales en GATM

Maintenant que nous nous sommes familiarisés avec les collocations, leur importance en GAT et les fonctions lexicales, présentons quelques méthodologies de l'implémentation des FL en GATM. Lambrey (2016, p. 59) reprend trois problématiques principales de l'implémentation de FL en GAT, comme abordé par Iordanskaja, Kim & Polguère (1996, p. 280) : « quand faut-il intégrer les FL dans un système de TAL? Est-ce que leurs noms doivent figurer en RSyntP ou bien leurs valeurs directement ? Comment doivent-elles être encodées dans les ressources employées par le système de TAL? ».

Comparons maintenant quelques méthodologies d'implémentation des FL appliquées en GATM.

Heid & Raab (1989) sont parmi les premiers à utiliser les fonctions lexicales dans la GATM basée sur la Théorie Sens-Texte. Dans le cadre du projet Polygloss (GATM en français et en allemand), ils proposent l'heuristique suivante :

> *First the basis is lexicalised, then the collocate, depending on which lexeme has been chosen as the basis.*

Selon les auteurs, les collocations peuvent être classifiées selon les catégories syntaxiques des arguments et des prédicats (par exemple, quand l'argument est un nom, les prédicats possibles sont des noms, verbes ou adjectifs). Ainsi, dans le dictionnaire de Polygloss, chaque entrée de lexème contient des emplacements (« slots ») pour les fonctions lexicales dont les remplissages (« fillers ») sont des collocatifs possibles (cf. Figure 4).

(problem

(...)

(caus func (create, pose))

(real (solve, ...))

(...))

Figure 4 Structuration d'une entrée du dictionnaire (Heid & Raab, 1989)

Le choix de la fonction lexicale pour un lexème donné est fait à partir de la classe sémantique à laquelle le lexème appartient. Autrement dit, l'application et le comportement des FL dépend des propriétés sémantiques de la base :

> *It seems possible to generalize over some regularities in collocation formation for members of semantically homogenous classes.*

Les auteurs illustrent cette généralisation en donnant un exemple des noms qui expriment l'information enregistrée dans l'ordinateur *répertoire, fichier, message* : ils possedent les mêmes valeurs des FL (= *supprimer*) en français et en allemand. Les exceptions doivent être enregistrées dans le dictionnaire sémantique.

En ce qui concerne les propriétés syntaxiques, elles sont accessibles en se référant au lexique de réalisation. Afin de pouvoir combiner les unités lexicales, il est donc nécessaire que le dictionnaire de lexicalisation soit modulaire :

> *The modular design of the lexicon supports generation of variants by giving access to all information needed at the appropriate choicepoints.*

De ce fait, trois dictionnaires sont proposés : sémantique, syntaxique et morphologique.

Selon Heid & Raab (1989), les FL doivent être intégrées dans les dictionnaires du système de GATM et à la fois dans le processus computationnel (règles guidant la lexicalisation). Après avoir investigué des tentatives de généraliser les FL en vue de la classe sémantique de la base, les auteurs se demandent aussi si les FL peuvent être généralisées d'après leur comportement syntaxique. Nous verrons dans le Chapitre 7 que le travail de Lambrey (2016) répond à cette question.

Le générateur de texte du projet Polygloss produit des textes en allemand et en français, langues qui restent assez similaires. Toutefois, il serait intéressant de voir si l'application des fonctions lexicales marcherait sur des langues typologiquement éloignées.

Lareau et al. (2011, 2012) proposent un système de GATM qui génère des commentaires de match de football dans deux langues radicalement différentes, l'anglais et l'arrernte (une langue australienne). Ils proposent une méthode d'encodage des fonctions lexicales en grammaire lexicale fonctionnelle (LFG) (Bresnan & Kaplan, 1982). À travers des exemples, les auteurs montrent que l'implémentation directe des fonctions lexicales au sein de la structure-f de la LFG n'est pas possible car celle-ci ne permet pas l'affectation de la valeur de l'attribut PRED sous forme de variable (par exemple, la structure *PRED : SV* au lieu de *PRED : kick* n'est pas permise).

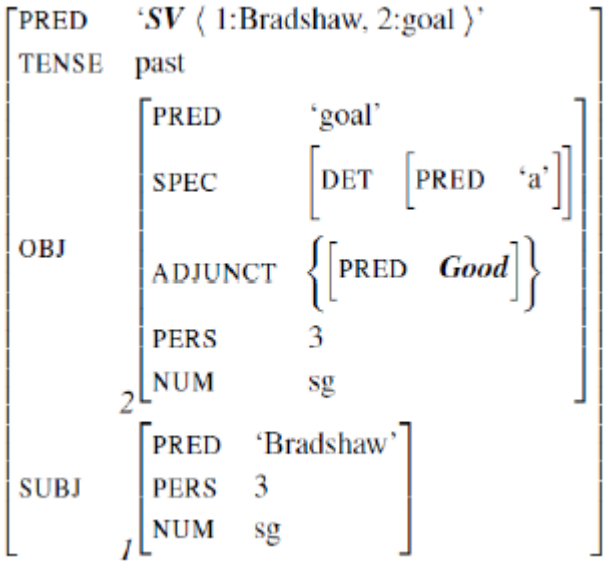


Figure 5 f-structure de « Bradshaw kicked a beautiful goal. » (Lareau et al., 2011)

Par conséquent, ils proposent d'encoder les FL au moyen de la « glue semantics » (Dalrymple, 2001) qui, selon les auteurs, offre plus d'expressivité pour traiter les correspondances complexes entre significations, lexèmes et leurs arguments. La « glue semantics » est basée sur la logique linéaire. En logique linéaire, les prémisses sont consommées lors de la déduction et, par conséquent, ne sont plus disponibles pour une preuve subséquente. Selon les auteurs, cela est particulièrement approprié lorsqu'il s'agît de construire le sens de la phrase : la contribution de chaque mot à la signification d'une phrase est unique et il ne doit pas y avoir de mots manquants ou redondants en termes de sens à être exprimé. Le système de GATM de Lareau et al. (2011, 2012) est alors basé sur le principe que chaque entrée lexicale de dictionnaire possède un constructeur sémantique qui établit le sens de la phrase à partir de ses composants. Voici un exemple de fonctionnement de la FL $Oper_1$ basé sur les contraintes logiques, tiré de (Lareau et al. 2011) :

goal N (↑PRED)='goal'

λX.goal(X) : ((OBJ↑)SUBJ) $_\sigma$ ⊸ ↑$_\sigma$

@OPER1(L)=

(↑ PRED)= '%stem⟨(↑ SUBJ),(↑ OBJ)⟩'

(↑ OBJ PRED)= c 'L'

λX.X : (↑ OBJ) σ ⊸↑σ

kick V @(OPER1 goal)

La base « goal » est un prédicat sémantique. Dans l'entrée de $Oper_1$, il est indiqué que la base du verbe doit être son objet syntaxique. Quand « kick » fait appel au patron « $Oper_1$ » avec l'argument « goal », la bonne structure syntaxique est construite.

Comme dans le projet de Polygloss (Heid & Raab,1989), le patron de $Oper_1$ est généralisé et, comme l'indique les auteurs, peut être appliqué à plusieurs langues.

Toutefois, les chercheurs repèrent quelques limites des FL. Tout d'abord, les FL ne sont pas toujours capables de traiter les divergences sémantiques entre les langues. Par exemple, il n'y a pas d'équivalent direct au verbe *teach* en arrernte : il faudrait dire *akaltye antheme* (littéralement *'*donner la connaissance'). L'entrée sémantique de ce concept en arrernte devrait donc être *cause (X, know (Y, Z)),* alors qu'en anglais l'entrée est *teach (X, Y, Z)*. Dans ce cas-là, l'application des fonctions lexicales est sans pertinence parce qu'il s'agît d'une différence conceptuelle plutôt que lexicale. Un autre problème qui se pose est le traitement des collocations non standard. La collocation *but gagnant* par exemple, dans laquelle *gagnant* signifie 'qui donne la victoire', est très spécifique et ne peut pas être réduite à un patron récurrent à travers les domaines et les langues. Les auteurs pensent que des FL « ad hoc » pourraient encore être définies pour de telles collocations, mais leur utilisation ne sera une solution viable que dans le contexte d'une application dans un domaine restreint.

Il important de remarquer que les applications de Heid & Raab (1989) et Lareau *et al.* (2011) sont partielles et ne consistent à traiter qu'un sous-ensemble de fonctions lexicales dans un domaine particulier. Le projet GÉCO s'intéresse à la méthodologie qui permettrait de modéliser les patrons des collocations de façon exhaustive dans un système générique de GATM. Dans la suite de cet état de l'art, nous nous concentrons sur le générateur GÉCO qui est le sujet du prochain chapitre.

Chapitre 4. GÉCO : successeur de MARQUIS

GÉCO (GÉnération de COllocations) (Lambrey & Lareau, 2015) est un générateur automatique de texte multilingue qui est en cours de développement à l'Université de Montréal. L'objectif de GÉCO est de décrire de façon exhaustive les patrons récurrents de collocations que l'on retrouve à travers les langues. Avant de décrire les spécificités de ce générateur, nous allons tout d'abord présenter le générateur MARQUIS (Wanner et al., 2010), dont est issu GÉCO.

1. MARQUIS

MARQUIS (Multimodal Air Quality Information Service for General Public) (Lareau & Wanner, 2007 ; Wanner et al., 2009, 2010) est un générateur de bulletins de qualité de l'air destiné au grand public et au personnel médical.

Selon Wanner et al. (2010), la qualité de l'air est un des problèmes centraux de la politique d'information environnementale dans le monde entier. En général, les services de transmission d'information de qualité de l'air ne présentent que les données brutes sous forme de graphiques, tableaux ou pictogrammes. Cependant, ces modes de présentation sont peu explicatifs et ne sont pas adaptés aux besoins des utilisateurs individuels. Comme l'indiquent Wanner et al. (2010), la création de MARQUIS avait pour objectif de fournir un service de génération d'information sur la qualité de l'air et de produire des bulletins multilingues individualisés sur les principaux polluants atmosphériques dans cinq régions européennes et leur effet sur la santé.

Les caractéristiques de MARQUIS les plus innovantes sont :

1. Optique orientée utilisateur : chaque bulletin fournit à l'utilisateur un contenu individualisé ;
2. Moyens de diffusion variés : internet, e-mail, mobile, SMS et WAP ;
3. Couverture des principaux polluants atmosphériques de chaque région ;
4. Techniques avancées de prévision de qualité de l'air ;

MARQUIS couvre huit langues européennes : le catalan, l'anglais, le français, le finnois, l'allemand, le polonais, l'espagnol et le portugais. Son architecture est divisée en deux parties principales. La réalisation linguistique est basée sur les principes de la théorie Sens-Texte et s'appuie sur des dictionnaires riches. Le générateur est développé sur la plateforme MATE, un transducteur de graphes libre (Bohnet & Wanner, 2010).

1.1 Architecture

L'architecture de MARQUIS est multistratale et comporte deux modules principaux : le traitement des données et la gestion des requêtes personnalisées. Regardons brièvement le fonctionnement de ces deux niveaux en reprenant (Wanner et al., 2010).

Dans une première étape, les données (concentrations de polluants) sont collectées, analysées et interprétées par le module d'évaluation de la qualité de l'air (AQAIM).

Dans la deuxième étape de traitement, le module de planification de document est déclenché par une demande d'information utilisateur. Une fois qu'une demande est détectée par l'interface utilisateur de MARQUIS, le planificateur de document reçoit le profil de l'utilisateur et la structure de sortie AQAIM pour la période en question (également déterminée par l'utilisateur). Le système sélectionne les connaissances correspondantes à partir de la structure de sortie d'AQAIM et de la base de connaissances et produit un plan de document contenant les unités d'information nécessaires pour composer le texte. Une fois que le plan de document est prêt, il est envoyé au module de génération de surface, qui produit le texte final. Ce dernier est livré à l'utilisateur via l'interface client.

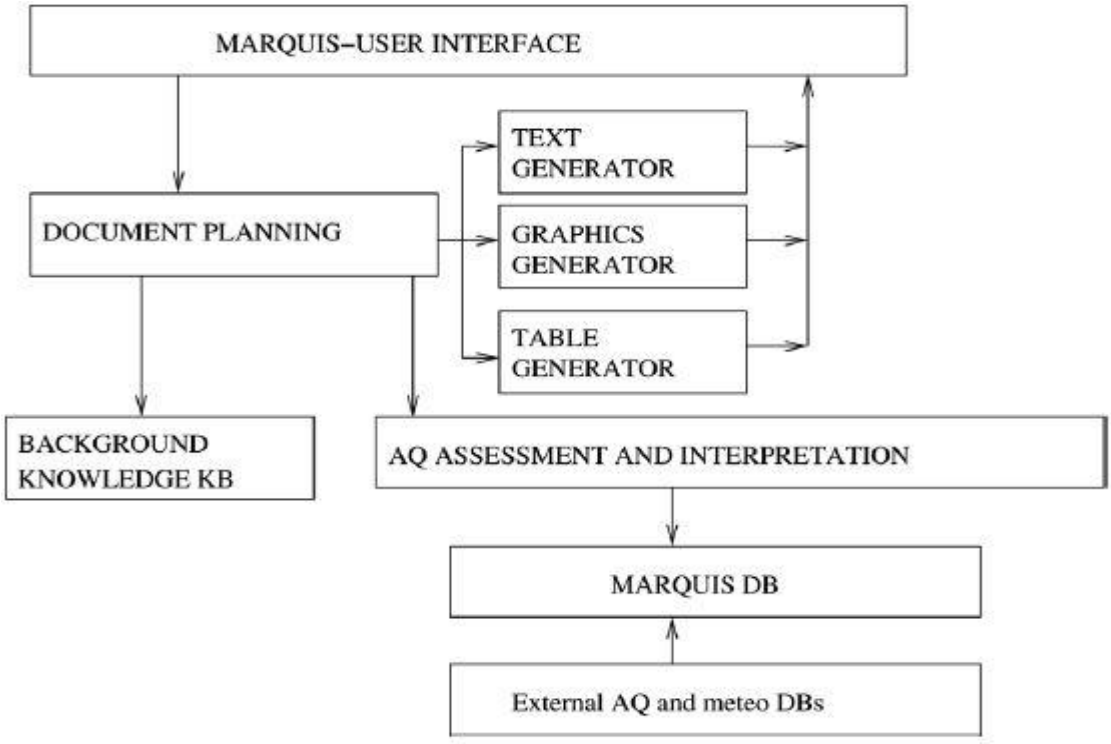


Figure 6 Architecture de MARQUIS (Wanner et al.,2010)

Pour résumer, l'organisation de MARQUIS suit plus ou moins « l'architecture de consensus » définie par Reiter & Dale (2000) dont nous avons parlé au Chapitre 1. La Figure 4 illustre graphiquement l'architecture de MARQUIS.

Le module de MARQUIS sur lequel nous nous focalisons dans ce mémoire est le réalisateur de surface. C'est ce que nous allons maintenant développer.

1.2 MARQUIS et la Théorie Sens–Texte

Dans MARQUIS, la réalisation linguistique est basée sur la Théorie Sens-Texte (Mel'čuk, 1967, 1997). Dans les lignes qui suivent, nous donnons une introduction à la Théorie Sens-Texte. Nous nous basons essentiellement sur les articles et ouvrages suivants : (Mel'čuk, 1993), (Mel'čuk, 1997), (Polguère, 1998a, 1998b), (Kahane, 2001a), (Steinlin 2003).

1.2.1 Cadre théorique

La Théorie Sens-Texte (TST) fait partie de la mouvance des grammaires formelles et propose une partition de la modélisation des énoncés en niveaux de représentation sémantique, syntaxique, morphologique et phonologique. La correspondance sens-texte est envisagée sous l'angle de la synthèse (du sens vers le texte), plutôt que de l'analyse (du texte vers le sens). Mel'čuk (1997) présente les trois postulats de la TST :

Premièrement, la langue est « un système fini de règles qui spécifient une correspondance multi-multivoque entre l'ensemble infini dénombrable de sens et un ensemble infini dénombrable de textes » (Mel'čuk, 1997).

Deuxièmement, la correspondance Sens-Texte « doit être décrite par un dispositif logique, qui constitue un modèle fonctionnel de la langue de type Sens-Texte ; il doit être élaboré et présenté dans la direction Sens ⇒ Texte ». Le modèle Sens-Texte suit le parcours onomasiologique : il est organisé « à partir du sens vers le texte, c'est-à-dire dans le sens de la synthèse, ou de production de la parole — plutôt que dans le sens opposé, celui de l'analyse, ou de la compréhension de la parole » (Mel'čuk, 1997). Autrement dit, la TST cherche à modéliser l'activité langagière du locuteur.

Troisièmement, pour naviguer du sens vers le texte, « deux niveaux intermédiaires de représentation des énoncés sont nécessaires pour mettre en lumière les faits linguistiques pertinents : la représentation syntaxique [= RSynt], qui correspond aux régularités

spécifiques à la phrase, et la représentation morphologique [= RMorph], qui correspond aux régularités spécifiques au mot » (Mel'čuk, 1997). Chaque niveau de représentation est divisé en niveaux profond et de surface. Le niveau profond est orienté vers le sens, tandis que le niveau surface est orienté vers le texte.

D'un point de vue fonctionnel, la correspondance Sens-Texte s'effectue au moyen de la construction de modèles formels. Polguère (1998) illustre ces modèles comme des machines logiques virtuelles du type suivant :

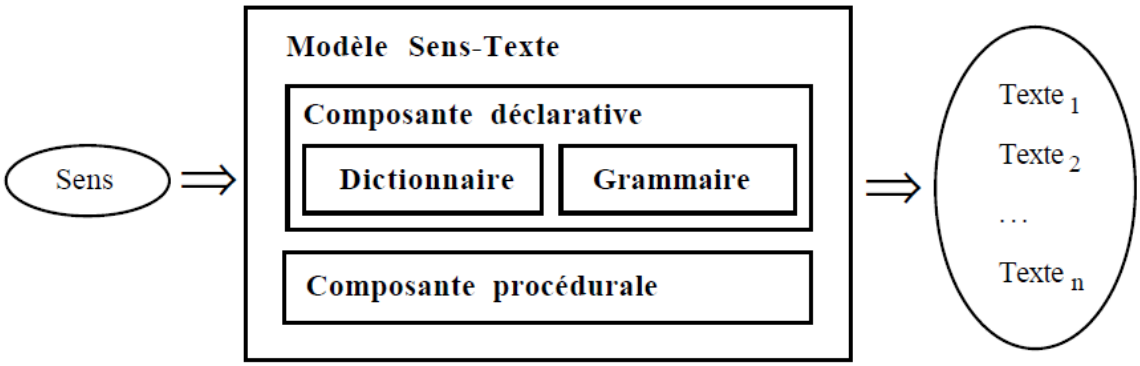


Figure 7 Modèle Sens-Texte (Polguère, 1998)

Toutes les composantes du modèle Sens-Texte — le dictionnaire et la grammaire — doivent être bien « accordées », puisqu'elles sont destinées à « collaborer » étroitement au cours du processus de synthèse des textes (Mel'čuk, 1997). La Figure 7 (Polguère, 1998) illustre le fonctionnement de le machine virtuelle Sens-Texte, qui prend en entrée un sens et retourne en sortie un ensemble de textes. Les textes contiennent toutes les paraphrases permettant d'exprimer le Sens donné en entrée. Cela met en valeur une des propriétés du langage naturel : sa capacité de paraphrasage, considérée comme le principe fondamental de la théorie Sens-Texte (Mel'čuk, 1997) :

> *Le modèle Sens-Texte procède d'une représentation du sens qui n'est pas autre chose qu'une représentation formelle de l'invariant d'un ensemble de paraphrases (plus ou moins) synonymes. A partir d'une représentation sémantique, le modèle Sens-Texte produit l'ensemble des phrases porteuses du sens correspondant à cette représentation sémantique ; le mécanisme clé qui assure cette production est appelé SYSTEME DE PARAPHRASAGE, et il fait partie de la composante sémantique du modèle.*

Ainsi, pour donner un exemple, le modèle Sens-Texte du français permettrait d'effectuer la correspondance suivante (tiré de Polguère, 1998) :

Sens	**Textes**
'Norm aime sa femme Marge de façon très intense'	*Norm aime follement sa femme Marge.* *Norm aime sa femme Marge à la folie.* *Norm aime sa femme Marge comme un fou.* *Norm éprouve un amour fou pour sa femme Marge.* *Norm ressent un amour immense pour sa femme Marge.*

Il est important de souligner que la TST suppose l'existence d'un lexique riche, celui-ci étant décrit de la façon la plus exhaustive que possible. Dans cette perspective, comme le souligne Mel'čuk (1997), l'importance du lexique est telle qu'on pourrait qualifier la TST de « lexicocentrique ». Le dictionnaire postulé par la TST est formalisé et s'appelle Dictionnaire Explicatif et Combinatoire (DEC). Comme l'indique son titre, le dictionnaire est explicatif car il fournit sur les lexies une définition sémantique : « il décrit le sens de l'unité lexicale au moyen d'une définition analytique qui décompose ce sens en termes de sens plus simples (c'est-à-dire, de sens qui sont plus proches des primitifs sémantiques que le sens de départ) » (Polguère, 1998). Le DEC est combinatoire, car il décrit la combinatoire syntaxique (sous-catégorisation) et lexicale (collocations) associée à chaque lexie.

Le système MARQUIS est basé sur l'implémentation de dictionnaires riches qui sera décrite de façon détaillée dans la section 1.4.1 . Nous allons maintenant nous concentrer sur les niveaux de représentation sémantique et syntaxique de la TST ainsi que les règles de correspondance d'un niveau à l'autre.

1.2.2 Les représentations sémantiques et syntaxiques

Dans la section précédente, nous avons vu que la correspondance Sens – Texte s'effectue à travers des représentations intermédiaires : sémantique, syntaxique, morphologique et phonologique. Le point de départ est une représentation du sens sous forme de réseau sémantique. Les niveaux RSyntP et RSyntS sont des arbres de dépendance

tandis que le niveau RMorph est une chaîne de morphèmes. Dans ce mémoire, nous nous intéressons aux représentations sémantiques et syntaxiques.

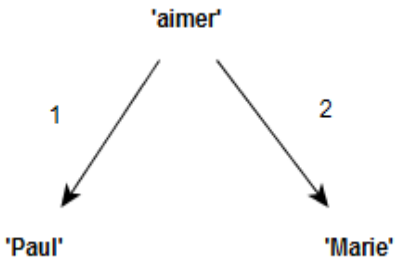


Figure 8 Exemple de structure sémantique

Comme illustré dans la Figure 8 ci-dessus, la **structure sémantique** est un graphe acyclique orienté. Les nœuds sont étiquetés par des sémantèmes. Tous les actants d'un sémantème doivent être représentés par des arcs qui portent des numéros permettant de les distinguer. Il est important de souligner que les nœuds du réseau sémantique sont des sémantèmes et non des unités lexicales. Ainsi, grâce aux informations encodées dans le dictionnaire, le sémantème ‘aimer’ peut se lexicaliser par exemple par AIMER, AMOUR, ou AMOUREUX.

La **représentation syntaxique** est synthétisée à partir de la représentation sémantique. Elle ressemble à l’approche syntaxique de Tesnière (1965) et est représentée formellement par un arbre de dépendance ; la structure syntaxique d'une phrase est l'ensemble des liens de dépendance entre les mots de la phrase. Il existe deux niveaux de représentation syntaxique : la représentation syntaxique profonde (RSyntP) et la représentation syntaxique de surface (RSyntS).

La **RSyntP** correspond à l’arborisation de la structure sémantique. Les nœuds de l'arbre sont étiquetés par des lexies profondes (lexies pleines ou fonctions lexicales) et les arcs sont étiquetés par des relations syntaxiques profondes. La RSyntP de la Figure 9 contient les deux relations actantielles syntaxiques profondes I et II, qui sont les pendants syntaxiques des relations actantielles sémantiques 1 et 2 : premier et deuxième actant du prédicat ‘aimer’. Une autre relation syntaxique profonde apparaissant dans la Figure 9 est la relation attributive ATTR, qui factorise tous les types de modificateurs. La Figure 9 illustre des arborisations possibles du réseau de la figure 8 en syntaxe profonde.

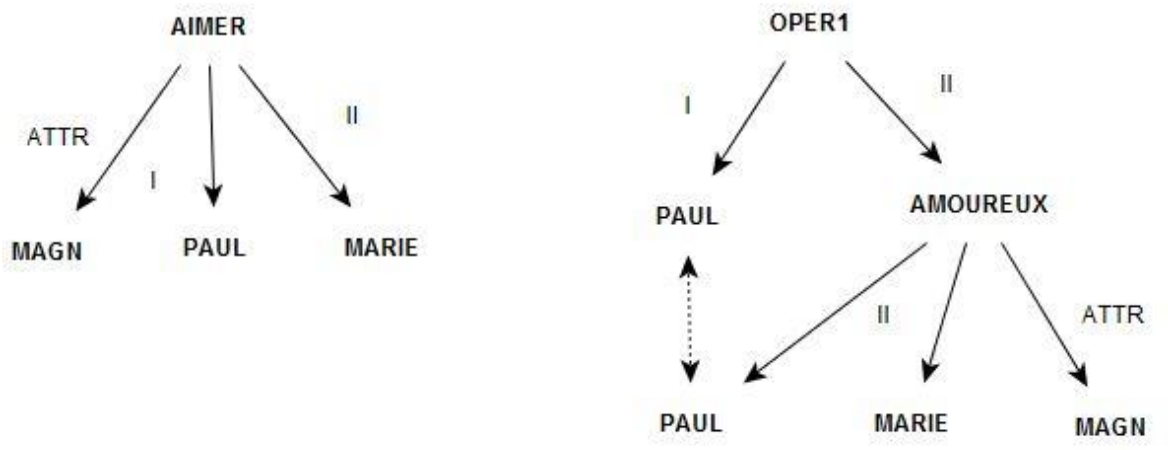


Figure 9 Arborisations possibles en RSyntP

La **RSyntS** est générée à partir des dépendances syntaxiques profondes et représente une ou plusieurs représentations syntaxiques de surface sous forme d'un arbre de dépendance. Les nœuds de l'arbre représentent des lexies de surface contenant des lexies vides (mots grammaticaux). Les arcs sont étiquetés par des relations syntaxiques de surface : les trois relations actantielles syntaxiques profondes I, II et ATTR, qui sont transformées en relations subjectale, objectale et modificative. Cette structure présente également les valeurs possibles des fonctions lexicales présentes dans la RSyntP. La Figure 10 donne une arborisation possible du réseau en syntaxe de surface.

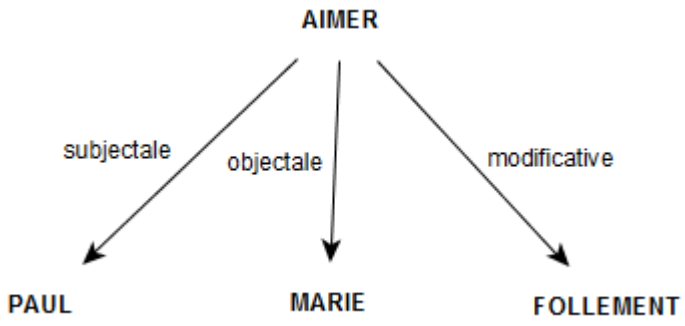


Figure 10 Arborisation en RSyntS

Les correspondances entre les représentations de deux niveaux adjacents s'effectuent grâce à des règles de correspondance. Les règles sémantiques assurent la correspondance entre la RSém et la RSyntP. La correspondance entre la RSyntP et la RSyntS se fait selon

les règles de syntaxe profonde. Les règles s'appuient sur les informations lexicales encodées dans les dictionnaires.

Nous venons de présenter le cadre théorique, les représentations syntaxique et sémantique et les règles de correspondance de la théorie Sens – Texte. Cela est une présentation partielle, car elle ne couvre pas les représentations morphologique et phonologique. Pour des informations plus complètes, le lecteur est invité à se référer à (Mel'čuk, 1997 ; Polguère 1998a). Maintenant que nous connaissons les principes de la théorie Sens-Texte, nous pouvons aborder la réalisation linguistique de MARQUIS.

1.3 MARQUIS et réalisation linguistique

Comme nous l'avons déjà vu, la réalisation linguistique de MARQUIS est basée sur le modèle multistratal de la théorie Sens-Texte. Le mécanisme de génération de surface de MARQUIS est divisé en six niveaux de représentation (cf. Figure 11) : sémantique (graphes orientés acycliques représentant des prédicats sémantiques et leurs actants), syntaxique profonde et de surface (arbres de dépendance), morphologique profond et de surface (structure linéarisée) et textuel (message final). Comme le processus de génération s'effectue à partir d'une représentation de nature non linguistique (données numériques), un niveau conceptuel supplémentaire est ajouté.

Dans MARQUIS, le passage d'un niveau à l'autre se fait par des transductions entre les règles de correspondance (de niveau i à niveau adjacent i+1) composant la grammaire de la langue, ainsi que sur différents dictionnaires. Le générateur part d'une structure conceptuelle qui est commune à toutes les langues en question. Le reste du processus de génération est adapté à chaque langue. Pourtant, les règles de transduction sont génériques et partagées entre toutes les langues, puisqu'elles récupèrent les informations encodées dans des dictionnaires riches. L'ensemble des règles de grammaire et des dictionnaires est implémenté sur la plateforme de transduction de graphes MATE (Bohnet & Wanner, 2010), que nous allons maintenant présenter.

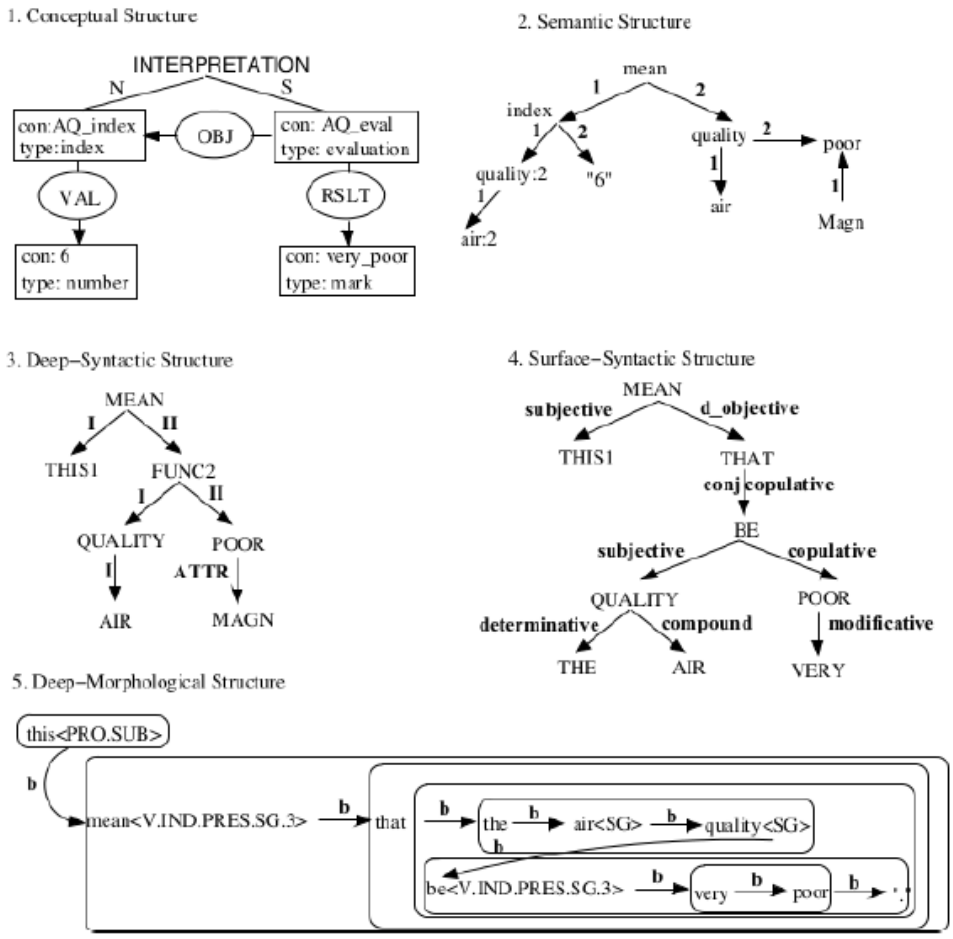


Figure 11 Mécanisme de génération de surface de MARQUIS (Wanner et al., 2010)

1.4 MATE

La plateforme MATE (Bohnet, 2006 ; Bohnet et al., 2000, 2007; Bohnet & Wanner, 2010) est un transducteur de graphe libre qui a été créé pour manipuler les structures de la théorie Sens-Texte. Ce transducteur construit, à partir d'un graphe donné en entrée, un ou plusieurs graphe(s) correspondant(s) d'un niveau adjacent, de façon non destructive. Le formalisme de transducteur de graphe et l'environnement de développement présenté ci-dessus ont déjà été utilisés notamment dans les projets de GAT MARQUIS, PATExpert et PESCaDO.

MATE se compose d'un certain nombre de modules de développement de dictionnaires et grammaires. Le générateur est basé sur un transducteur qui mappe toute structure d'entrée spécifiée automatiquement ou manuellement S_{ij} de la strate i sur sa structure équivalente $S_i{+}1_k$ de la strate i+1 en appliquant le module de grammaire correspondant $G_i \Leftrightarrow i{+}1$ à S_{ij} selon un ou plusieurs dictionnaires donné(s).

1.4.1 Dictionnaires

Les dictionnaires de MATE contiennent deux types d'information : le vocabulaire et des informations concernant la correspondance entre les éléments du vocabulaire de strates adjacentes. Dans MARQUIS, il existe trois dictionnaires principaux : conceptuel, sémantique et lexical. Chaque séquence de transduction récupère l'information contenue dans ces dictionnaires. Maintenant, nous allons décrire rapidement chacun de ces dictionnaires en reprenant les exemples de Lareau & Wanner (2007).

Le **dictionnaire conceptuel** est utilisé pour encoder des concepts et leurs correspondants sémantiques dans les différentes langues. Comme illustré dans l'exemple ci-dessous, le concept concentration est une instance de la classe property_attribute. Dans cet exemple, « concentration » est renvoyé au prédicat sémantique correspondant, 'concentration'. À partir de cette entrée, MATE est capable de sémanticiser ce concept en un prédicat sémantique.

```
concentration : property_attribute {
  sem = concentration
}
```

Le **dictionnaire sémantique** fournit, pour chaque sémantème décrit, toutes ses lexicalisations possibles dans une langue donnée lors du passage de la structure sémantique vers la structure syntaxique profonde d'un énoncé.

```
concentration {
  lex = concentration
}
augmentation {
  lex = augmenter
  lex = augmentation
}
```

Le **lexicon** contient, pour chaque unité lexicale, des informations caractéristiques grammaticales, sa valence et ses fonctions lexicales. Ces trois types d'information nous renseignent sur ses propriétés combinatoires.

```
gp = {
// Sem-Dsynt valency projection (1=>I, 2=>II):
1 = I
2 = II
// First syntactic actant can be realized as « ozone
concentration »:
I = {
dpos = N
rel = compound
det = no
}
// First syntactic actant can be realized as « concentration
of ozone »:
I = {
dpos = N
rel = noun_completive
prep = to
det = no
}
// Second syntactic actant can be realized as « concentration
of 180ug/m3 »:
II = {
dpos = Num
rel = noun_completive
prep = of
}
}
// (iii) Lexical functions:
Magn = high
AntiMagn = low
Adv1 = in // « (we found) ozone in a concentration (of 180
ug/m3) »
Func2 = be // « the concentration (of ozone) is 180 ug/m3 »
Oper1 = have // « ozone has a concentration (of 180 ug/m³) »
IncepFunc2 = reach // « the concentration (of ozone) reached
180 ug/m3 »
IncepOper1 = reach //ozone will reach a concentration of 180
ug/m3 »
}
```

MATE permet d'encoder autant de dictionnaires que nécessaire. Selon Lareau & Wanner (2007), les dictionnaires sont essentiels au processus du module de génération de surface de MARQUIS car l'implémentation de dictionnaires riches permet de concevoir des règles grammaticales plus génériques qui peuvent être utilisées pour plusieurs langues de différentes familles.

1.4.2 Grammaires

Afin de décrire le fonctionnement de grammaire de MATE, nous nous appuyons ici sur les explications simplifiées de (Lambrey, 2016 p. 78-80). Nous avons décidé d'illustrer le fonctionnement en prenant un exemple facilement compréhensible. Pour la description formelle du fonctionnement de l'outil, le lecteur est invité à se référer à (Bohnet, 2010 ; Wanner 2010).

Comme nous l'avons dit précédemment, la réalisation linguistique de MARQUIS s'effectue à partir d'un graphe conceptuel. A travers de plusieurs étapes de transduction, MATE crée la structure textuelle qui correspond au graphe de départ. Ainsi, les règles de transduction constituent la grammaire de la langue.

Dans MATE, l'ensemble de règles de grammaire est divisé en trois parties : le côté gauche, le côté droit et les conditions d'application. Le côté gauche représente la structure de départ : les nœuds et les arcs pour l'application de la règle. L côté droit représente le graphe de sortie dans le niveau adjacent ou autrement dit, les nœuds et les arcs crées à partir de graphe source. La troisième partie, les conditions d'application limitent l'application des règles. Le tableau 1 (Lambrey, 2016) illustre le fonctionnement de règle de lexicalisation simple de RSem - RSyntP de MARQUIS.

Partie Gauche	**Patrie Droite**

`l:?Xl{}`
`?L <- semanticon::(?Xl.sem).(lex)`
	`rc:?Xr{ <=> ?Xl`
	`dlex=?L`
	`}`

Conditions d'application	
`semanticon::(?Xl.sem).lex;`	

Tableau 1 Règle de lexicalisation dans MATE (Lambrey, 2016)

Brièvement, la règle illustrée dans cet exemple récupère un nœud (quelconque) de la RSem et crée son correspondant lexicalisé dans la RSyntP. Dans ce qui suit, nous expliquons la composition et l'application des règles présentes dans chacune des parties.

Partie gauche :

- **?Xl** : un noeud à lexicaliser

- **{}** : arguments du noeud ne sont pas précisés
- **l:** : le noeud est bloqué, ce qui rend ?Xl indisponible pour d'autres règles
- **?L** : variable qui prend pour valeur le résultat de la commande semanticon::(?Xl.sem).(lex).
- **semanticon::(?Xl.sem).(lex).** : commande qui permet à fouiller le semanticon et chercher l'entrée pour le sémantème du noeud ?Xl,

Partie droite :

- **rc:** : le contexte de droite
- **?Xr. <=>?Xl** : ?Xr est le correspondant du noeud ?Xl
- **dlex = ?L** : création d'un attribut dlex dont la valeur est la lexicalisation de ?Xl.

Condition d'application :

- **semanticon::(?Xl.sem).lex;** : condition vérifiant que le sémantème ?Xl possède bien une lexicalisation dans le semanticon.

MATE permet de créer autant de règles de grammaire que nécessaire. Les règles créées sont rassemblées dans les différents modules représentant chaque interface de transition, par exemple sémantique vers syntaxe profonde, syntaxe profonde vers syntaxe de surface, etc. Dans chaque module, les règles sont regroupées en paquets, ce qui permet le partage des conditions d'application parmi les règles de chaque paquet. Il est important de souligner que les règles sont créées de façon le plus générique possible afin de pouvoir les appliquer pour plusieurs langues.

1.4.3 Aspect multilingue

L'architecture modulaire de MATE permet le partage des règles de grammaire selon le principe de *grammar sharing* que nous avons présenté dans le Chapitre 2. (Lareau & Wanner, 2007) soulignent qu'une telle conception du système appliquée dans le projet MARQUIS a permis d'ajouter rapidement de nouvelles langues typologiquement similaires au générateur car peu de changements ont dû être apportés aux règles de grammaire, puisque la plupart des informations dépendantes de la langue sont enregistrées dans les dictionnaires.

Comme illustré dans la Figure 12, plus de 85% des règles de chaque module de MARQUIS sont génériques et le pourcentage de règles spécifiques n'est pas plus que 16%.

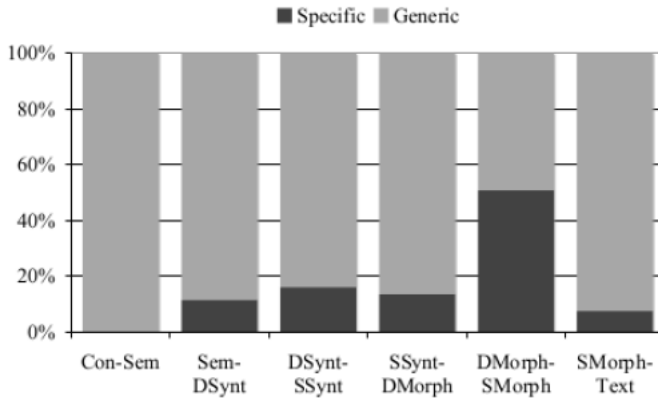


Figure 12 Statistiques de règles génériques et spécifiques dans chaque module de MARQUIS

Maintenant que nous nous sommes familiarisés avec l'organisation la réalisation linguistique de MARQUIS et la plateforme MATE, nous allons présenter le générateur automatique de texte GÉCO.

2 GÉCO

Dans cette section, nous présentons GÉCO (GÉnérateur de COllocations), le générateur de texte multilingue qui est en cours de développement à l'Université de Montréal. L'objectif principal de GÉCO (Lambrey & Lareau, 2015 ; Lambrey, 2016), est de décrire de façon exhaustive les patrons des collocations et de pouvoir générer les collocations en plusieurs langues. GÉCO est basé sur l'architecture et la grammaire de MARQUIS et est implémenté sur la plateforme MATE.

Dans GÉCO, les collocations sont modélisées à l'aide de fonctions lexicales. Au total, 26 259 fonctions lexicales différentes ont été implémentées via 129 règles de transduction. Selon (Lambrey, 2016), ces règles sont valables théoriquement pour plusieurs langues. Dans cette section, nous ne nous intéressons pas aux techniques précises de l'implémentation des collocations. Elles seront expliquées en détail dans la partie XX de notre mémoire. Afin de nous familiariser avec GÉCO, nous parcourons rapidement les conclusions du mémoire de (Lambrey, 2016 p. 111).

Dans GÉCO, la représentation et le fonctionnement des FL ont été modélisés sous forme de règles de correspondance entre le niveau sémantique **RSem** et le niveau syntaxique profond **RSyntP**. Le passage s'effectue à l'aide des règles de transduction qui forment la grammaire de GÉCO.

En plus de règles de transduction, GÉCO s'appuie sur les informations encodées dans trois dictionnaires : le **semanticon**, contenant les sémantèmes, le **lexicon** contenant toutes les entrées lexicales et le dictionnaire de fonctions lexicales, **lf** qui inclut les informations sur les fonctions lexicales et leur fonctionnement en représentation syntaxique profonde. Les informations encodées dans ces trois dictionnaires sont récupérées par les règles de transduction afin de lexicaliser les unités lexicales.

Chaque fonction lexicale standard syntagmatique simple et complexe a été modélisée (le processus de modélisation sera présenté dans le Chapitre 7). Ainsi, il s'est avéré que plusieurs fonctions lexicales présentent un comportement similaire en RSem et RSyntP. Alors, plutôt que de créer une règle de transduction pour chaque FL, elles ont été regroupées dans des patrons génériques. Au total, 10 patrons ont été créés. À chaque patron correspond une règle de transduction. Par la suite, les fonctions lexicales standard syntagmatiques complexes ont été intégrées.

Quoique le fonctionnement de GÉCO s'appuie sur les principes de la théorie Sens-Texte, quelques ajustements ont dû être effectués : afin de généraliser le comportement de certaines FL, leur composante sémantique a été intégrée directement dans RSem. Aussi, nous avons inclus les valeurs des fonctions lexicales en RSyntP alors que la théorie prévoit de ne les introduire qu'en RSyntS.

L'évaluation du système a été effectuée en calculant en comparant le répertoire avec ceux de DiCoInfo, DiCoEnviro et DiCoLiLex : « GÉCO couvre la moitié des fonctions lexicales standard syntagmatiques simples et complexes de ces ressources, ce qui se traduit par une couverture de 85 % des occurrences de ces fonctions lexicales. Autrement dit, nous couvrons les fonctions lexicales qui sont les plus représentées dans ces ressources, telles que Magn ou $Oper_1$. De ce fait, notre inventaire de fonctions lexicales est exhaustif dans le sens où il couvre la grande majorité des occurrences de fonctions lexicales » (Lambrey, 2016 p. 151).

La prochaine étape du projet de GÉCO est d'introduire plusieurs modules de génération multilingue et de tester la performance du système. Dans le cadre de notre mémoire, nous allons introduire un module de génération du lituanien et nous allons générer des collocations dans cette langue.

Conclusion

Dans le premier chapitre de notre état de l'art, nous avons parcouru les principes fondamentaux de GAT : l'entrée et la sortie, l'architecture modulaire ainsi que les tâches principales que le système de GAT doit effectuer afin de générer un texte cohérent et compréhensible pour un humain.

Ensuite, nous avons pris connaissance des spécificités de système de génération multilingue. Il s'est avéré que la plupart des systèmes sont fondés sur le principe de partage de ressources linguistiques. Nous avons décrit deux méthodes de partage de ressources : l'adaptation de grammaires et le partage de grammaires. Nous avons vu que la première approche est plutôt adaptée pour les langues typologiquement similaires, car elle vise à modifier une grammaire déjà existante pour couvrir une nouvelle langue. La deuxième approche est capable de couvrir une grande variété de langues : les règles sont partagées entre les langues plutôt que simplement recyclées. Après avoir comparé quelques projets de GATM fondés sur cette dernière approche, nous avons conclu que la conception d'un système multilingue à base de règles exige une modélisation des phénomènes linguistiques abstraits qui sont communs à plusieurs langues et qui peuvent constituer la base pour les modules spécifiques de chaque langue du système.

Dans le troisième chapitre, nous nous sommes familiarisés avec le processus de la lexicalisation et les problèmes liés à celui-ci : les concepts ne peuvent toujours pas être directement liés aux lexèmes et ces derniers influencent l'apparition et les positions des autres lexèmes dans la phrase. Nous avons vu que certaines cooccurrences lexicales sont souvent restreintes du point de vue sémantique et syntaxique et sont modélisées via les fonctions lexicales. Les fonctions lexicales standard sont universelles : elles existent dans toutes les langues et suffisent à décrire l'ensemble des collocations de façon systématique et formelle. Les collocations peuvent former des patrons communs à plusieurs langues. Elles sont donc adaptées au traitement multilingue et peuvent nous servir à modéliser les phénomènes linguistiques génériques et indépendants de la langue. C'est ce que nous chercherons à implémenter dans un système de GATM basé sur le partage de ressources.

Les fonctions lexicales ont déjà été implémentées en GATM. Après avoir comparé quelques projets, nous avons vu qu'il est possible d'implémenter les fonctions lexicales à différents niveaux du processus de traitement linguistique. Ainsi que nous l'avons constaté,

l'application des FL dans ces projets était partielle et ne consistait à traiter qu'un sous-ensemble des fonctions lexicales. Par conséquent, la question de la généralisation des FL est ouverte : comment peut-on modéliser les patrons des collocations de façon exhaustive et de les intégrer dans un système générique de GATM ?

Le projet GÉCO, qui est le sujet du dernier chapitre, modélise le comportement des FL sous des patrons génériques. GÉCO est un générateur automatique de texte multilingue exploitant des ressources linguistiques riches fondées sur le principe du partage de grammaires. Il est basé sur l'architecture multistratale de MARQUIS et le formalisme de la Théorie Sens-Texte. Le générateur est implémenté sur la plateforme MATE, un transducteur de graphes libre. L'ensemble des règles de transduction forme la grammaire de GÉCO. Dans ce système, la représentation et le fonctionnement des FL ont été modélisés entre les niveaux sémantique et syntaxique profond. En plus de règles de transduction, GÉCO s'appuie sur l'implémentation de dictionnaires riches.

L'objectif de notre stage était d'intégrer un module de génération du lituanien dans GÉCO et de générer des collocations. Il est important de souligner que dans le cadre de ce stage, nous visions à générer des arbres syntaxiques dans le niveau représentation syntaxique de surface de la Théorie Sens-Texte sur laquelle est basé GÉCO. Afin d'atteindre cet objectif, nous nous étions fixé les tâches suivantes :

- Trouver un corpus contenant plusieurs collocations
- Repérer les collocations dans le corpus et les modéliser via les fonctions lexicales
- Représenter les phrases du corpus sous forme de réseaux sémantiques
- Créer les arbres syntaxiques attendues
- Intégrer les réseaux sémantiques dans MATE
- Créer un dictionnaire de lexicalisation dans MATE
- Reprendre les règles de grammaire de MARQUIS et les appliquer au lituanien, créer de nouvelles règles si les règles de MARQUIS ne sont pas capables de couvrir nos besoins
- Évaluer le résultat obtenu

Partie 2

-

Ajout du lituanien à GÉCO

La deuxième partie de ce mémoire présente l'ensemble des tâches que nous avons effectuées afin d'introduire un module du lituanien dans GÉCO. Cette partie s'articule autour de quatre chapitres. Dans le Chapitre 5, nous présentons le lituanien, le corpus auquel nous nous référons pour générér les phrases, et les phénomènes linguistiques que nous devions prendre en compte. Dans le Chapitre 6, nous décrivons les changements que nous avons effectués dans les dictionnaires afin de traiter certains phénomènes lexicaux et grammaticaux. Le Chaptire 7 est destiné à présenter le traitement des collocations du lituanien. Enfin, dans le Chapitre 8, nous présentons les nouvelles règles de grammaire que nous avons créées dans GÉCO.

Chapitre 5. Corpus

Comme indiqué dans (Reiter & Dale, 2000, pp. 31-33), un élément clé du design d'un système de GAT est la détermination des entrées qui seront fournies au système et des textes de sortie que celui-ci devrait produire. La détermination des entrées et des sorties d'un système de GAT repose généralement sur une collection d'exemples d'entrées et de texte associés. Cette collection de données d'entrée et de sortie s'appelle un corpus. En général, le corpus est créé en utilisant des textes écrits par les humains et il est basé sur de véritables rapports. Il devrait couvrir toute la gamme des textes attendus par le système de GAT. Celui-ci devrait être capable de gérer des variations dans les données saisies, et cela peut être difficile de déterminer quelle sera la gamme complète des textes générés. Comme le précisent les auteurs, ce qui compte comme corpus représentatif est ouvert au débat et doit être convenu d'un accord commun par les développeurs et les utilisateurs.

Dans le cadre de notre stage, nous avons décidé de nous baser sur texte suivi. L'entrée de notre système sont les représentations sémantiques de chacune des phrases de ce texte, et la sortie attendu est l'ensemble des structures syntaxiques correspondantes. L'objectif de ce chapitre est de présenter le corpus du lituanien que nous avons choisi comme modèle et de donner l'analyse linguistique des phrases. Nous donnons également les représentations syntaxiques que nous visons à générer à la fin de notre projet. Avant de commencer, nous introduisons brièvement le lituanien. Pour une description plus détaillée, le lecteur intéressé peut se référer à www.lituanus.org.

1. Le lituanien

Le lituanien est une langue indo-européenne faisant partie des langues baltes (avec le letton). D'un point de vue linguistique, le lituanien est souvent considéré comme la plus archaïque des langues indo-européennes modernes car il a conservé de nombreuses anciennes formes lexicales et grammaticales qui n'ont été attestées que dans d'autres langues indo-européennes éteintes telles que le latin, le grec et le sanskrit. Paul Thieme (1958) a illustré ce phénomène en comparant le proverbe lituanien *Dievas davė dantis; Dievas duos ir duonos* et sa traduction latine, *Deus dedit dentes; Deus dabit et panem* ('Dieu a donné des dents, Dieu donnera aussi du pain'), avec sa forme originale en sanskrit : *Devas adadā t datas; Devas dā t (dadā t) api dhā nā s*. Nous pouvons donner d'autres exemples, comme les mots lituaniens *sūnus* ('fils'), *vyras* ('homme'), *ratas* ('cercle'), *vėjas* ('vent'), *dūmas* ('fumée'), *sapnas* ('rêve') et les équivalents en sanscrit *sunus*, *viras*, *ratha*, *vajus*, *dhumas*, *svapnas* qui n'ont pas changé leurs formes pour les cinq mille dernières années. Parmi les langues parlées de nos jours, la langue lituanienne est probablement la plus proche du proto-indo-européen car elle a préservé de nombreux phénomènes propres à celui-ci.

Afin d'illustrer les aspects de la grammaire du lituanien, nous allons décrire brièvement les noms, les adjectifs et les verbes.

1.1 Noms

Le lituanien distingue nom propre et nom commun. Les noms sont variables en genre (masculin et féminin) et en nombre (singulier et pluriel). Il existe cinq déclinaisons. Elles sont définies par la flexion au singulier du nominatif et du génitif. Il existe sept cas : nominatif, génitif, datif, accusatif, instrumental, locatif et vocatif.

1.2.Adjectifs

Comme les noms, les adjectifs ont deux nombres (singulier et pluriel). Contrairement aux noms, ils ont 3 genres (masculin, féminin et neutre). Seuls les adjectifs de genre masculin et féminin s'accordent avec le nom (en nombre et en cas).

1.3.Verbes

Les catégories morphologiques et sémantiques des verbes lituaniens sont les suivantes :

- La réflexivité morphologique : verbes simples ou réflexifs.
- La transitivité : verbes transitifs ou intransitifs.

- La causativité : causatifs ou non causatifs.
- L'aspect : perfectif ou imperfectif.
- Le mode : l'indicatif, le conditionnel, l'impératif et le relatif.
- Le temps : quatre temps morphologiques (le présent, le prétérit, le passé itératif, le futur) et sept temps composés (3 temps inchoatifs et 4 temps antérieurs).
- La personne : 5 personnes de conjugaison.
- La voix : active ou passive.

Tous les verbes lituaniens possèdent trois conjugaisons. La conjugaison est définie selon la voyelle finale de la racine de la troisième personne du présent.

2. Le corpus

L'objectif principal de GÉCO étant la génération des collocations, nous avons cherché un type de texte qui contient, en général, plusieurs collocations. Nous avons ainsi choisi d'utiliser un texte de nouvelles judiciaires, ce type journalistique étant souvent très riche en collocations. Pour trouver un texte caractéristique de ce genre, nous avons parcouru la rubrique des nouvelles judiciaires d'un site de nouvelles populaire en Lituanie www.delfi.lt. Nous avons choisi un article selon deux critères : la taille de l'article (longueur moyenne) et le nombre de collocations présentes dans le texte. L'article tiré du http://www.delfi.lt/news/daily/crime/pasigailejo-tv-pagalbos-zurnalistes-zudiko-i-laisve-dar-iseis.d?id=72761580 a ainsi constitué notre corpus. Le corpus (ci-dessous) se compose de 16 phrases et de 177 mots. La traduction de chaque phrase est disponible dans la prochaine section.

> *Žurnalistę Viltę Stankutę ir jos draugą nužudęs Nerijus Kalaušis už grotų praleis 20 metų. Taip penktadienį nusprendė Lietuvos apeliacinis teismas, atmetęs nužudytosios žurnalistės artimųjų prašymą žudiką įkalinti iki gyvos galvos. Dėl dvigubos žmogžudystės 20 metų laisvės atėmimo bausme nuteistas N. Kalaušis taip pat teismo prašė jam dar labiau sušvelninti bausmę, bet ir šis prašymas nebuvo patenkintas. Vis dėlto, pasak teismo atstovės Vilmos Budėnienės, teisėjų kolegija pakeitė nuosprendį, konstatuodama, kad nužudymo metu nužudytoji V. Stankutė nebuvo bejėgiškos būklės. Pasak teismo, nors nukentėjusioji įvykio metu buvo apsvaigusi, bet tai nereiškia, jog dėl to ji buvo bejėgiškos būsenos. Tokią išvadą apeliacinės instancijos teismo teisėjų kolegija padarė įvertinusi bylos aplinkybes ir ekspertų paaiškinimus teisme. Tuo pačiu*

teisėjų kolegija nurodė, kad ši aplinkybė neturi įtakos bausmės dydžiui. Anot teismo, N. Kalaušis pirmuosius 5 metus turi praleisti Lukiškių kalėjime, o kitus – pataisos namuose. Be to, jis privalės sumokėti 165 tūkst. Eur neturtinės ir 4 tūkst. Eur turtinės žalos atlyginimą. Nužudymo byla teisme išnagrinėta neviešuose posėdžiuose. Lietuvos apeliacinio teismo paskelbta nutartis įsiteisėjo iš karto nuo jos paskelbimo, nors ją dar galima skųsti kasacine tvarka Lietuvos Aukščiausiajam Teismui. Daugybę kartų teistas N. Kalaušis kraupų nusikaltimą įvykdė 2014 metų naktį į gruodžio 27-ąją – tuomet Giraitėje (Kauno r.) rasti 31 metų žurnalistės V. Stankutės ir jos draugo G. Pavasario palaikai. N. Kalaušis buvo sulaikytas kitą dieną po įvykio. Vyras teisinosi, kad baisų nusikaltimą įvykdė apimtas pavydo. Tuo metu nužudytosios V. Stankutės sesuo L. Stankutė anksčiau yra sakiusi, kad sesuo kurį laiką bandė nutraukti ryšius su ūmaus būdo N. Kalaušiu, tačiau nuolat sulaukė grasinimų. Teigiama, kad N. Kalaušis grasino susidoroti, padegti namus, į kuriuos jį gyventi priėmė V. Stankutė ir kuriuose ji vėliau buvo nužudyta. Kuomet galiausiai žurnalistė vyrą išprašė iš namų, šis savo žiaurius grasinimus įvykdė.

3. Glosage du corpus

Nous avons traduit les phrases et nous avons glosé chaque mot avec son équivalent en français. Cela a été fait pour que l'équipe de GÉCO puisse se familiariser avec le lituanien et analyser le corpus. La traduction des phrases a également constitué une partie de la documentation de GÉCO pour les futurs utilisateurs.

Nous présentons les phrases du corpus en utilisant les règles de Leipzig pour les gloses (https://www.eva.mpg.de/lingua/resources/glossing-rules.php).

(1)

Žurnalistę	Viltę	Stankutę	ir	jos	draugą	nužudęs
journaliste-ACC.SG	Vilte.ACC	Stankute.ACC	et	elle-GEN.SG	ami-ACC.SG	tuer-PTCP-PST.NOM

Nerijus	Kalaušis	už	grotų	praleis	20	metų
Nerijus.NOM	Kalausis.NOM	derrière	barreau-GEN.PL	passer.FUT	20	an-GEN.PL

'Ayant tué la journaliste Viltė Stankutė et son compagnon, Nerijus Kalaušus passera 20 ans derrière les barreaux'

(2)

Taip	penktadienį	nusprendė	Lietuvos	apeliacinis	teismas,
Ainsi	vendredi-ACC	décider-PST-SG	Lituanie-GEN	appel-ADJ-NOM.SG	court-NOM.SG

atmetęs	nužudytosios	žurnalistės	artimųjų	prašymą
rejeter-PTCP-PST	PTCP-PASS	journaliste-GEN.SG	proches-GEN.PL	demande-ACC.SG

žudiką	įkalinti	iki	gyvos	galvos
tueur-GEN.SG	imprisoner-INF	jusqu'à	vivante-ADJ-GEN.SG.PL	tête-GEN.SG

'Ainsi vendredi a décidé la Cour d'appel de Lituanie, rejetant la demande des proches de la journaliste assassinée d'emprisonner le tueur à vie'

(3) Dėl dvigubos žmogžudystės 20 laisvės atėmimo
A cause de double-ADJ-GEN.SG meurtre-GEN.SG 20 liberté-GEN.SG deprivation-GEN.SG

bausme nuteistas N. Kalaušis taip pat teismo
peine-INST.SG condamner-PTCP-PST.NOM N. Kalaušis aussi court.GEN.SG

prašė jam dar labiau sušvelninti bausmę, bet
demander-PAST.SG il-DAT.SG encore plus réduire-INF peine-ACC.SG mais

ir šis prašymas nebuvo patenkintas.
et ce-NOM.SG demande-NOM.SG être-PST-NEG approuver-PTCP-PST.NOM

‘En raison du double meurtre d'une peine de 20 ans de prison condamné N. Kalaušis a demandé au tribunal de réduire davantage la peine, mais cela été refusé.’

(4) Vis pasak teismo atstovės Vilmos Budrėnienės, teisėjų
Toutefois selon court-GEN.SG représentante-GEN. Vilma.GEN Budreniene.GEN juge.GEN.PL

kolegija pakeitė nuosprendį konstatuodama, kad nužudym
collège.NOM.SG changer-PST.SG décision-ACC.SG GER que meurtre-GEN.SG

metu nužudytoji V. Stankutė nebuvo
temps.INSTR.SG assassiner-PTCP.PST.NOM V. Stankutė être-PST.SG-NEG

bejėgiškos būklės.
impuissant-ADJ-GEN.SG état-GEN.SG

‘Toutefois, selon la représentante du tribunal Vilmos Budėnienė, la Chambre de première instance a changé l'arrêt, constatant qu'au moment du meurtre, V. Stankutė n'était pas dans un état impuissant.’

(5) Pasak teismo nors nukentėjusioji įvykio metu
selon court-GEN.SG quoique victime- NOM.SG accident-GEN.SG temps-INSTR.SG

buvo apsvaigusi, bet tai nereiškia, jog dėl
être-PST.SG intoxiquer-PTCP.PST.NOM mais cela signifier-NEG.PRS.SGque à cause de

to ji Stankutę buvo bejėgiškos būklės.
cela.GEN elle.NOM.SG Stankute.ACC être-PST.SG impuissant-ADJ.GEN.SG état.GEN.SG

‘Selon le tribunal, bien que la victime ait été intoxiquée pendant l'événement, mais cela ne signifie pas qu'elle était dans un état impuissant.’

(6) Tokią išvadą apeliacinės instancijos teismo
Telle.ACC.SG conclusion-ACC.SG appel.ADJ.GEN.SG instance.GEN.SG cour.GEN.SG

kolegija padarė įvertinusi bylos aplinkybes ir
collège.NOM.SG faire-PST.SG évaluer-PTCP.PST cas-GEN.SG fait-ACC.PL et

ekspertų paaiškinimus teisme.
expert-GEN.PL explanation-ACC.PL court-LOC.SG

‘Cette conclusion a été faite par le tribunal de première instance de la cour d'appel après avoir évalué les faits et les explications des experts devant les tribunaux.’

(7) Tuo pačiu, teisėjų kolegija nurodė, kad ši aplinkybė
En même temps juges.GEN.PL collège.NOM.SG préciser-PST.SG que ce.NOM-SG fait-NOM.SG

neturi įtakos bausmės dydžiui.
avoir-INF-NEG influence-GEN.SG peine-GEN.SG taille-DAT.SG

‘Dans le même temps, le jury a déclaré que ce fait n'affecte pas la taille de la peine.’

(8) Anot teismo, N. Kalaušis pirmuosius 5 metus turi
Selon court-GEN.SG N. Kalaušis premier.ACC.PL 5 an-ACC.PL devoir-PRS.SG

praleisti Lukiškių kalėjime o kitus - pataisos
passer-INF Lukiškės-GEN prison-LOC.SG et autre.ACC.PL - correction-GEN.SG

namuose.
maison-LOC.SG

‘Selon le tribunal, N. Kalaušis doit passer les 5 premières années dans la prison de Lukiškės, et les autres dans la maison de correction.’

(9) Be to, jis privalės sumokėti 165 tūkst. Eur
En plus, il.NOM.SG devoir.FUT payer-INF 165 mille Eur

neturtinės ir 4 tūkst. turtinės žalos
moral-ADJ-GEN.SG et 4 mille matériel-ADJ.GEN.SG dommage-GEN.SG

atlyginimą.
compensation.ACC.SG

‘En outre, il doit payer une compensation de 165 000 euros de dommage moral et 4 000 euros de dommage matériel.’

(10) Nužudymo byla teisme išnagrinėta neviešuose
Meurtre-GEN.SG cas-NOM.SG court-LOC.SG examiner- PTCP.PST publique-ADJ-LOC.PL

posėdžiuose.
séance-LOC.PL

‘Le cas a été examiné dans les sessions publiques.’

(11) Lietuvos apeliacinio teismo paskelbta nutartis
Lituanie.GEN.SG appel-ADJ-GEN.SG court.GEN.SG publier-PTCP.PST décision-NOM.SG

įsiteisėjo iš karto jos paskelbimo, nors ją dar
se lėgaliser.PST tout de suite elle.GEN.SG publication-GEN.SG 20 bien que elle.ACC.SG encore

galima skųsti kasacine tvarka Lietuvos
pouvoir faire appel-INF cassation-ADJ-INS.SG ordre.INSTR.SG Lituanie.GEN.SG

Aukščiausiajam	Teismui.
suprème.DAT.SG	court.DAT.SG

‘La décision de la Cour d'appel de Lituanie est entrée en vigueur immédiatement après sa publication, mais il est encore cassation possible de faire appel à la Cour suprême de Lituanie’

(12)

Daugybę	kartų	teistas	N.	Kalaušis	kraupų
Plusieurs	fois-GEN.PL	condamner-PTCP.PST	N.	Kalaušis	terrible-ADJ-ACC.SG

nusikaltimą	įvykdė	2014	metų	naktį	į
crime-ACC.SG	commetre-PST.SG	2014	an-GEN.PL	NUIT-ACC.SG	à

gruodžio	27-ąją	- tuomet	Giraitėje	Kauno r.	rasti
décembre.GEN.SG	27	- quand	Giraite.LOC	Kaunas.GEN	trouver-PTCP.PST

31	metų	žurnalistės	V.	Stankutės	ir	jos
31	an-GEN.PL	journaliste.GEN.SG	V.	Stankutė.GEN	ir	elle.GEN.SG

draugo	G.	Pavasario	palaikai.
ami.GEN.SG	G.	Pavasaris.GEN	reste.NOM.PL

‘Plusieurs fois condamné N. Kalaušis terrible crime a commis en 2014, la nuit du 27 décembre à Giraitė (région de Kaunas) où ont été retrouvés les restes du journaliste de 31 ans V. Stankutė et son compagnon G.Pavasaris.’

(13)

Vyras	teisinosi,	kad	baisų	nusikaltimą	įvykdė
Homme.NOM.SG	expliquer-PST.SG	que	terrible-ADJ ACC.SG	crime-ACC.SG	commetre-PST.SG

apimtas	pavydo
envahir-PTCP.PST	jalousie-GEN.SG

‘L'homme a expliqué qu'il a commis un crime terrible à cause de la jalousie.’

(14)

Tuo	metu	nužudytosios	V.	Stankutės	sesuo
Ce-INSTR.SG	temps-INS.SG	assassinée-GEN.SG	V.	Stankutė.GEN	soeur-NOM.SG

L.	Stankutė	yra	sakiusi	kad	sesuo	kurį
L.	Stankutė.NOM	être-PRS.SG	dire.PTCP.PST	que	soeur-NOM.SG	quelque-ACC.SG

laiką	bandė	nutraukti	ryšius	su	ūmaus	N.
temps.ACC.SG	essayer-PST.SG	rompre.INF	relation-ACC.PL	avec	aigu-ADJ.GEN.SG	N.

Kalaušiu	tačiau	nuolat	saulaukė	grasinimų.
Kalaušis.INSTR	mais	régulièrement	reçevoir-PST.SG	menace-GEN.PL

‘À l'époque, la sœur de l’assassinée V. L. Stankutė , L. Stankutė a déclaré que sa sœur tentait de rompre avec le N. Kalaušis, mais a régulièrement reçu des menaces.’

(15)

Teigiama	kad	N.	Kalaušis	grasino	susidoroti,	padegti
Indiquer-PCTP.PRS	que	N.	Kalaušis	menacer-PST.SG	battre-INF	incendier-INF

namus	į	kuriuos	jį	gyventi	priėmė	V.
maison-ACC.PL	dans	laquel.LOC.PL	il-ACC.SG	vivre.INF	accepter-PST.SG	V.

Stankutė	ir	kuriuose	ji	buvo	nužudyta
Stankutė.NOM	et	lequel-LOC.PL	elle-NOM.SG	être-PST.SG	tuer-PCTP.PST

'Il est indiqué que N. Kalaušis a menacé de tuer, de mettre le feu à la maison dans laquelle V. Stankutė l'a invité à vivre et où elle a été tuée par la suite.'

(16)

Kuomet	galiausiai	žurnalistė	vyrą	išprašė	iš
Quand	enfin	journaliste-NOM.SG	homme.ACC.SG	rejeter-PST.SG	de

namų	šis	savo	žiaurius	grasinimus	įvykdė.
maison-GEN.SG	il-NOM.SG	son	terrible-ADJ ACC.PL	menace-ACC.PL	mener-PST.SG

'Lorsque la journaliste a finalement rejetté l'homme de la maison, il a accompli ses terribles menaces.'

4. Phénomènes linguistiques à traiter

Selon (Reiter & Dale, 2000, pp. 31-33), une fois que le corpus est constitué, l'analyse des exigences pour le système peut commencer. L'objectif principal de l'analyse est de déterminer le contenu d'information dans le corpus. Il peut être analysé en termes d'éléments d'information qui correspondent aux objets, propriétés et autres variables dans un domaine particulier. Ces identifications sont exprimées linguistiquement au moyen des variantes de constructions grammaticales ou les phrases les plus typiques dans le corpus. L'analyse de contenu d'information nécessite donc de classer chaque phrase, clause ou autre constituant grammatical de corpus dans des catégories.

Dans le cadre de notre projet, nous nous intéressons aux collocations et aux phénomènes sémantiques et syntaxiques. L'étape suivante consiste donc à identifier les collocations, les fonctions lexicales et les caractéristiques grammaticales de construction des phrases.

Nous avons donc analysé chaque phrase du corpus et nous avons identifié les phénomènes linguistiques que nous devrons prendre en compte afin de pouvoir générer les arbres syntaxiques cibles. Dans les prochaines sections, nous décrivons les phénomènes que nous avons identifiés et les illustrons avec des exemples.

4.1.Collocations

Nous avons identifié quinze collocations.

Lituanien		**Français**	
Base	**Collocatif**	**Base**	**Collocatif**
nusikaltimas	*kraupus*	*crime*	*affreux*
prašymas	*atmesti*	*demande*	*rejeter*
prašymas	*patenkinti*	*demande*	*approuver*
nuosprendis	*keisti*	*décision*	*changer*
išvada	*padaryti*	*conclusion*	*faire*
byla	*nagrintėti*	*cas*	*analyser*
nutartis	*įsiteisinti*	*ordonnance*	*se légitimer*
nusikaltimas	*įvykdyti*	*crime*	*commettre*
ryšys	*nutraukti*	*relation*	*rompre*
grasinimas	*sulaukti*	*menace*	*recevoir*
namai	*išprašyti iš*	*maison*	*expulser*
grasinimas	*įvykdyti*	*menace*	*commettre*
grasinimas	*baisus*	*menace*	*terrible*
kolegija	*teisėjų*	*collège*	*juge*
pavydas	*apimti*	*jalousie*	*envahir*

4.2. Locutions

En plus des collocations, nous avons identifié **dix locutions** :

Trois locutions verbales

- *pasodinti už grotų* ‘mettre derrière les barreaux’ (lit. ‘asseoir derrière barreaux’)
- *skųsti kasacine tvarka* ‘faire appel en cassation’ (lit. ‘se plaindre en cassation’)
- *sukelti gaisrą* ‘mettre le feu’ (lit. ‘causer feu’)

Cinq locutions nominales

- *pataisos namai* ‘maison de correction’

- *Lietuvos apeliacinis teismas* ‘Cour d’appel de Lituanie’
- *apeliacinės instancijos teismo teisėjų kolegija* ‘le pannel des juges de la Cour d’appel’
- *kuris laikas* ‘quelque temps’
- *iki gyvos galvos* ‘à vie’ (lit. ‘jusqu’à tête vivante’)

Deux locutions adverbiales

- *be atvangos* ‘sans cesse’
- *iš karto* ‘tout de suite’ (lit. ‘de suite’)

4.3.Phrases complexes

Nous avons repéré plusieurs phrases complexes, c’est-à-dire des phrases qui comportent plus d’un verbe conjugué (plusieurs propositions). Notre système devra donc générer des **phrases coordonnées**, **phrases subordonnées** et **phrases relatives**.

Il est important de remarquer que les phrases du corpus sont longues et plusieurs comportent à la fois une coordination, une subordination et une relative. Notre système sera donc confronté à la génération de phrases très complexes.

4.4.Noms et cas

Nous avons repéré des noms communs et des noms propres. En ce qui concerne les cas, nous avons trouvé des occurrences de tous les cas du lituanien sauf le vocatif. Notre système devra donc générer le nombre et le cas approprié pour chaque nom de l’arbre syntaxique généré.

Il est important de souligner qu’il existe des cas strictement syntaxiques (ex. nominatif = sujet) alors que d’autres ont une valeur sémantique, comme le locatif et l’instrumental. On le voit en lituanien avec les structures du type *Jean (est) à la maison,* où *à la maison* sera réalisé par un nom portant le cas locatif. Nous avons observé ce phénomène dans le phrases 10 et 15 de notre corpus. Nous avons traité ce phénomène à l’aide de la FL Loc_{in}, qui réfère à l’entité où se passe l’action et se trouve donc dans les structures sémantiques.

4.5.Types des verbes et temps verbaux

L’article est écrit au passif et quelques phrases sont au futur.

4.6.Localisation temporelle

Dans le corpus, cinq phrases comportent un élément temporel sémantique, qui indique le temps d'un événement ou d'une action. Par exemple, dans la Phrase 2, il est indiqué que la cour a pris la décision vendredi. Dans la Phrase 4, deux actions prennent lieu en même temps. Afin de modéliser les phrases de ce type, nous avons introduit un sémantème générique 'Time' dans les structures sémantiques. Le premier actant du sémantème pointe vers l'action ou l'évenement à situer tandis que le deuxième est pointé vers le moment de l'action.

5. Création des structures sémantiques

Dans la deuxième étape du traitement, nous avons manuellement créé la représentation sémantique de chaque phrase du corpus. Dans l'approche de Théorie Sens-Texte, une représentation sémantique est un graphe orienté acyclique où les nœuds représentent les sémantèmes et les arcs représentent les relations prédicat-argument entre ces sémantèmes (Mel'čuk, 2013). Dans les prochains paragraphes, nous présentons les bases théoriques sur lesquelles nous nous sommes appuyée pour créer les réseaux sémantiques. Mais d'abord, il nous semble utile d'expliquer ici les notions de **sens liant**, **actant sémantique** et **position actancielle**.

Comme Polguère (2012) l'indique, la notion de **sens liant** a été proposée afin de particulariser les sémantèmes ayant la capacité de gouverner d'autres sémantèmes dans une structure sémantique. Autrement dit, il s'agit d'un sémantème qui contrôle des positions des actants sémantiques.

Pour ce qui est de la notion d'**actant sémantique**, Mel'čuk & Polguère (2008) proposent la définition suivante :

« Un actant sémantique d'une lexie L, dans une structure sémantique donnée, est un sémantème qui remplit une position actancielle sémantique associée à L dans le lexique ». Les auteurs définissent aussi la **position actancielle** :

« Une position actancielle sémantique associée à une lexie L dans le lexique est une variable sémantique (notée X, Y, Z…) qui remplit les deux conditions suivantes :

1) du point de vue conceptuel, elle correspond à un participant de la situation dénotée par L ;

2) du point de vue syntaxique, un sémantème qui occupe la position en question doit être exprimable dans la phrase sous le contrôle syntaxique de L ; cela signifie que l'on peut construire des phrases où l'expression d'un tel sémantème apparait soit comme un dépendant syntaxique direct de L, soit comme un dépendant syntaxique d'un collocatif verbal de L ».

Dans le même article, les auteurs évoquent la nature sémantique prédicative d'un sens et la présence des positions actancielles qu'il contrôle.

Un sens liant qui dénote un **fait** (action, activité, événement, état, etc.) est appelé **prédicat sémantique**. Selon Mel'čuk (2006), les prédicats sémantiques s'expriment par des lexies de toutes les parties du discours profondes : nom, verbe, adjectif, adverbe.

Un sens liant qui dénote une **entité** (objets physiques, substances, etc.) est appelé **quasi-prédicat sémantique**. Ces entités ont aussi des actants car elles contiennent un sens prédicatif en position dominante dans leur décomposition. Tels sont les significations des noms des artefacts (instruments, armes, véhicules, etc.), les noms des parties (de quelque chose), les noms des fonctions et institutions sociales, les noms relationnels tels que les liens de parenté, etc.

Un sens non liant qui dénote un fait est appelé un **prédicat non actanciel**. Les prédicats non actanciels sont des verbes à sujet vide, par exemple *pleuvoir* et *neiger*.

Enfin, un sens non liant qui dénote une entité est un **nom sémantique** : *zèbre*, *eau*, etc.

Selon Mel'čuk (2004), les significations prédictives peuvent être exprimées par des unités lexicales appartenant à toute partie du discours : verbes, adjectifs, adverbes, ainsi que toutes les prépositions et conjonctions, les nombres et des particules.

Pour créer les structures sémantiques nous nous sommes donc appuyée sur les bases théoriques que nous venons de présenter plus haut. Nous avons implémenté les structures dans MATE.

Il est important de noter que nous avons simplifié certaines phrases en retirant les éléments qui ne sont pas essentiels du point de vue sémantique. Les structures générées ne visent pas à reproduire exactement les phrases du corpus. Nous y référons pour reprendre leur sens principal et pour observer leur comportement grammatical et syntaxique.

Nous notons également que certaines FL font partie des structures sémantiques. Comme nous l'avons expliqué dans le Chapitre 4, un tel ajustement a dû être fait afin de

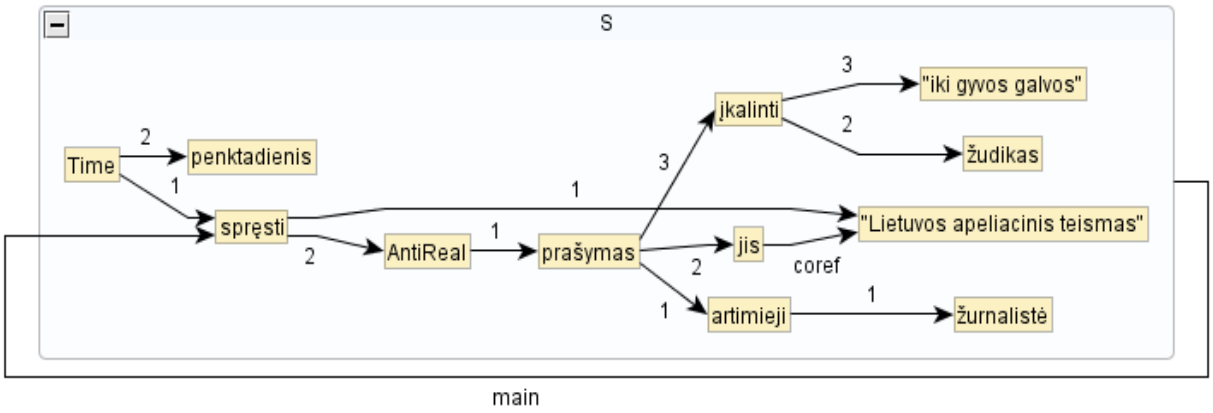
généraliser le comportement de certaines FL. Ce choix sera expliqué plus en détail dans le Chapitre 7. De même, deux autres composants ont été ajoutés dans les structures sémantiques : le sémantème ‘Time’, qui indique le temps d’un événement, et les anaphores (dans les structures sémantiques, *coref*) pour marquer les reprises sémantiques qui seront nécessaires dans l’arborisation en RSyntP. Ces choix seront expliqués plus en détail dans le Chapitre 8.

6. Structures sémantiques, structures syntaxiques cibles et structures syntaxiques produites

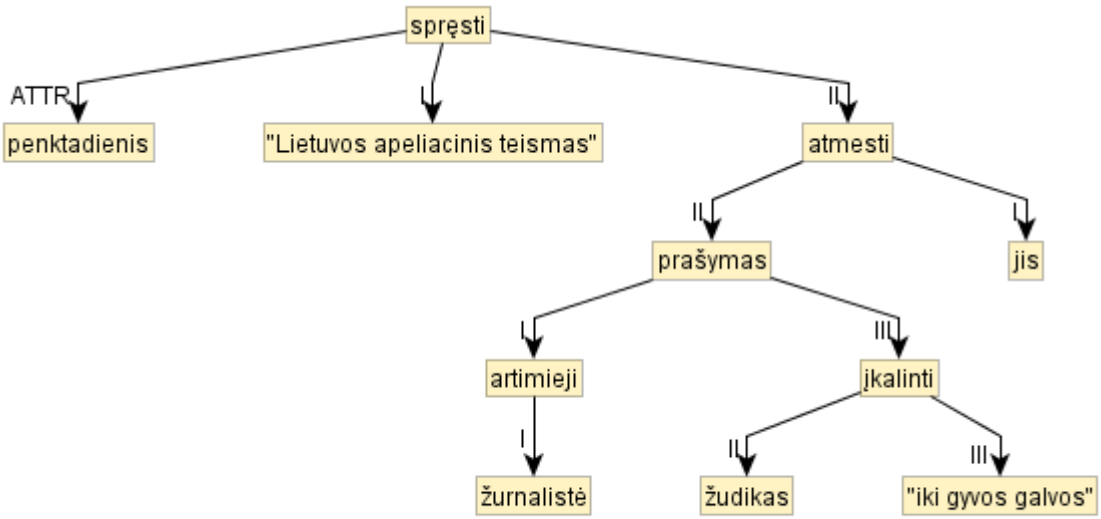
Après avoir représenté le sens des phrases sous forme de représentations sémantiques (RSém), nous avons construit des arbres syntaxiques en RSyntP que notre générateur devrait générer à partir de la RSém. Dans cette section, nous donnons un exemple d’une structure sémantique en RSém et sa structure syntaxique cible en RSyntP. Dans cette section, nous profitons également pour donner quelques exemples des structures produites en RSyntS. L’ensemble des structures sémantiques et structures syntaxiques produites sont disponibles dans le système de GÉCO à l’Université de Montréal, elles sont sous la disposition de François Lareau.

Voici les structures de la **Phrase 2** :

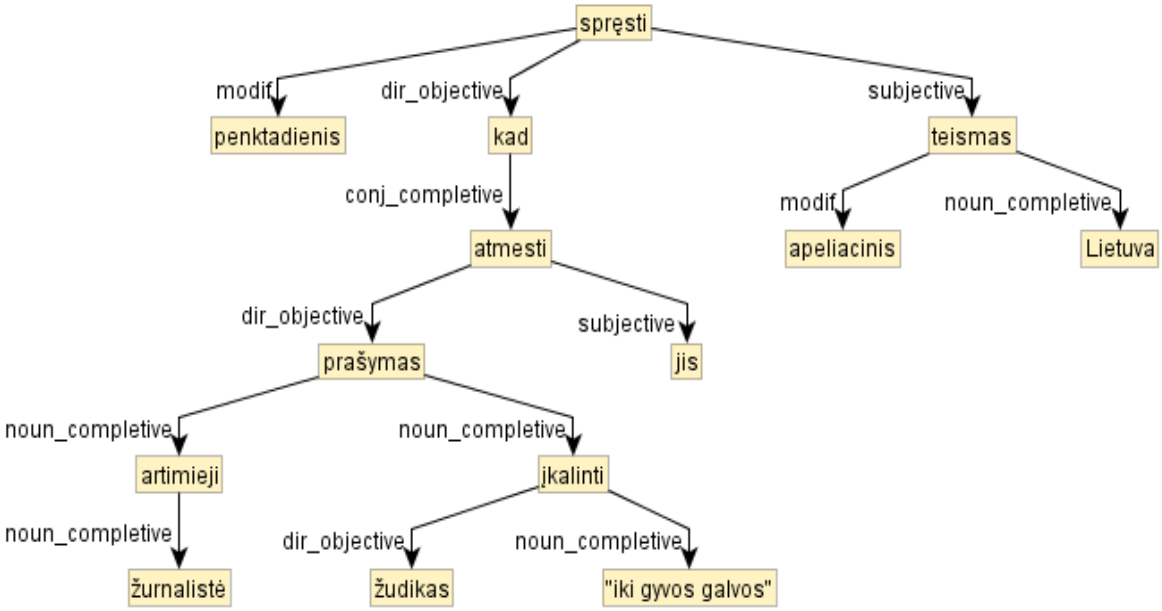
Structure sémantique (RSém)



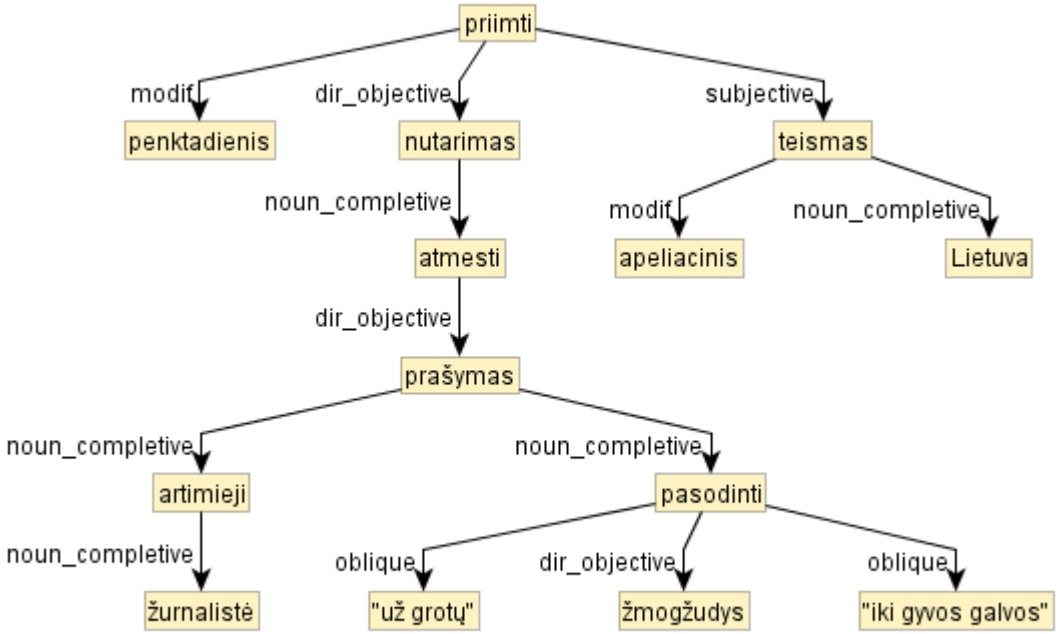
Structure syntaxique cible (RSyntP) :

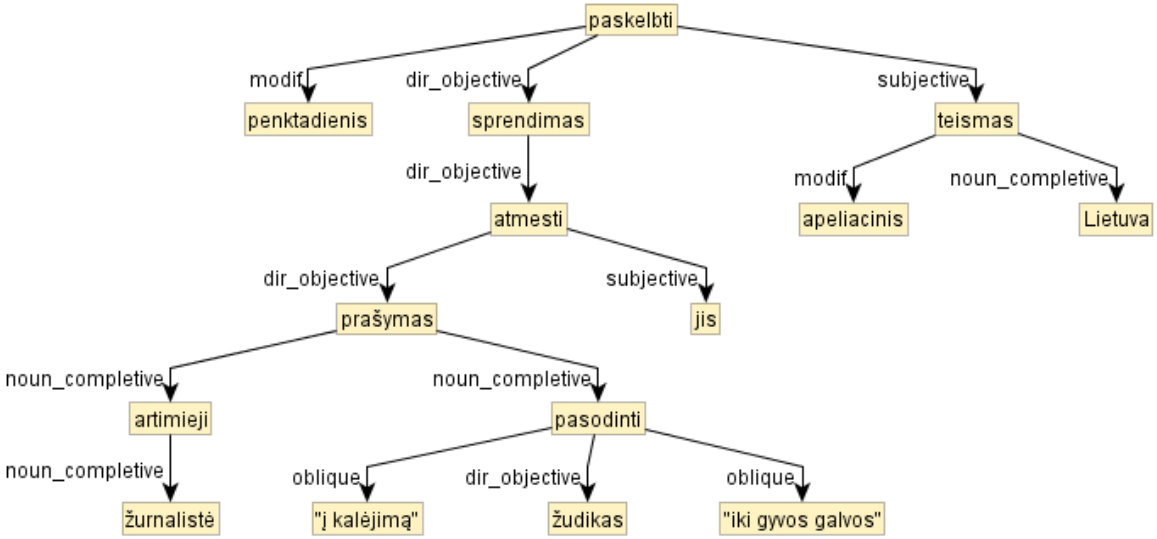


Structure syntaxique produite (RSyntS) :



Quelques paraphrases (RSyntS) :


priimti
modif
dir_objective
subjective
penktadienis
nutarimas
teismas
noun_completive
modif
noun_completive
atmesti
apeliacinis
Lietuva
dir_objective
prašymas
noun_completive
noun_completive
artimieji
pasodinti
noun_completive
oblique
dir_objective
oblique
žurnalistė
"už grotų"
žmogžudys
"iki gyvos galvos"


paskelbti
modif
dir_objective
subjective
penktadienis
sprendimas
teismas
dir_objective
modif
noun_completive
atmesti
apeliacinis
Lietuva
dir_objective
subjective
prašymas
jis
noun_completive
noun_completive
artimieji
pasodinti
noun_completive
oblique
dir_objective
oblique
žurnalistė
"į kalėjimą"
žudikas
"iki gyvos galvos"

Chapitre 6. Traitement des données linguistiques

Nous avons vu que l'algorithme de génération de GÉCO repose sur les principes de partage de grammaire développés par Lareau & Wanner (2007) pour le système MARQUIS, c'est à dire la conception de règles de grammaires suffisamment génériques pour pouvoir s'appliquer à plusieurs langues. Pour cela, le générateur repose sur des ressources lexicographiques riches. Regardons maintenant ce que nous avons développé dans les dictionnaires de GÉCO.

GÉCO utilise trois dictionnaires : *semanticon*, qui liste les sémantèmes d'une langue et donne leur lexicalisation, *lexicon*, qui donne les propriétés lexicales et syntaxiques des lexèmes de la langue, et *lf*, qui donne les caractéristiques sémantiques et syntaxiques des fonctions lexicales (ce fichier est partagé par toutes les langues).

Dans ***semanticon***, dictionnaire des sémantèmes, nous avons introduit chaque sémantème qui fait partie des structures sémantiques. Pour chacun, nous avons listé toutes les lexicalisations possibles. Prenons, par exemple l'entrée de 'spręsti' ('décider') :

```
spręsti {
        lex = sprendimas
        lex = spręsti
        lex = nuosprendis
        lex = nutarimas
        lex = nutarti
}
```

Pour ce sémantème, nous avons ajouté cinq lexicalisations possibles (*lex*) : *sprendimas, spręsti, nuosprendis, nutarimas, nutarti*. Le lexème *spręsti* est un verbe qui signifie 'décider'. Les lexèmes *sprendimas*, *nuosprendis* et *nutarimas* sont des synonymes nominaux de *spręsti* correspond au nom *décision*. Le dernier lexème est un verbe synonyme de *spręsti* 'décider'. Le sémantème 'spręsti' peut donc se lexicaliser par des lexèmes de deux catégories : nom et verbe. Au total, nous avons introduit X sémantèmes et X lexicalisations possibles dans le ***semanticon***.

Dans GÉCO, le ***lexicon*** est divisé en trois parties. Regardons quelles informations sont enregistrées dans ces trois parties et quels paramètres nous avons changés afin de décrire les lexèmes de notre corpus en lituanien.

1. Données métalinguistiques

Une entrée de dictionnaire META contient des informations sur la langue (nom et encodage) et les paramètres grammaticaux par défaut, tels que la définitude et le nombre des noms ainsi que le temps et le mode des verbes. Pour notre corpus, nous avons choisi 'indéfini' et 'singulier' comme valeurs par défaut pour les noms et 'passé' et 'indicatif' pour les verbes.

2. Attributs par défaut

Cette partie du dictionnaire modélise le régime des lexèmes, c'est-à-dire la réalisation des actants sémantiques en RSyntP et RSyntS. Chaque lexème dans le dictionnaire hérite d'une classe abstraite qui spécifie les valeurs par défaut de certains attributs. Nous allons maintenant les présenter.

La classe au sommet de la hiérarchie est **predicate**. Elle instancie un prédicat et les informations sur ses arguments. Le prédicat dans le dictionnaire fait référence à un nœud prédicatif donné dans la représentation sémantique de la phrase. Les informations sur ses arguments sont enregistrées dans le *gp* (patron de régime). Le *gp* fournit les informations sur la projection du régime sémantique au régime syntaxique.

```
predicate {
  gp = {
    1 = I
    2 = II
    3 = III
    4 = IV
    5 = V
    6 = VI
  }
}
```

Comme illustré dans l'exemple ci-dessus, un prédicat peut avoir au maximum six actants. Les actants sémantiques sont désignés par des chiffres arabes alors que les actants syntaxiques sont en chiffres romains. Ainsi, l'actant 1 en RSém se réalise par défaut comme l'actant I en RSyntP, l'actant 2 comme II et ainsi de suite. Bien sûr, un lexème particulier peut toujours dérroger à cette règle générale.

Les catégories grammaticales sont divisées en classes ayant un comportement prédicatif similaire. Par exemple, la classe **nom** prévoit jusqu'à trois arguments pour un nom

prédicatif. Les parties du discours qui héritent de cette classe (nom propre, pronom et pronom personnel) ont des propriétés linguistiques similaires. Illustrons cette organisation avec un exemple :

```
noun : predicate {
  dpos = N
  spos = noun
  countable = yes
  gp = {
    I = { rel = noun_completive
          case = GEN }
    II = { rel = noun_completive }
    III = { rel = noun_completive }
  }
```

Cette entrée définit la classe des noms dans le *lexicon*. Les caractéristiques grammaticales en RSyntP et RSyntS par défaut sont :

dpos = *N* : la partie du discours en RSyntP est un nom

spos = *noun* : la partie du discours RSyntS est un nom

Les informations sur le régime du nom commun sont disponibles dans le *gp*. Tous les actants (*I*, *II*, *III*) en RSyntP se réalisent par des compléments de nom (rel = *noun_completive*) en RSyntS. Il faut remarquer qu'en lituanien, le premier actant du nom est toujours au cas génitif, donc l'attribut *case* = *GEN* a été ajouté à l'actant I.

De cette manière, 4 **classes de prédicats** sont décrites. Chaque classe possède des sous-classes qui héritent de ses informations linguistiques et en précisent d'autres. Dans la liste ci-dessous, nous donnons la liste des classes et leurs sous-classes ainsi que les informations que nous avons ajoutées pour traiter le lituanien.

1. **Noms :** nom commun, nom propre, pronom, pronom personnel
2. **Verbes :**
    1. **verb** : verbes simples, l'actant I en RSyntP se réalise comme sujet en RSyntS (*rel* = *subjective*). En lituanien, le cas du sujet d'un verbe est toujours nominatif.
    2. **verb_dt** : verbes transitifs directs. L'actant II en RSyntP se réalise comme objet direct en RSyntS (*rel* = *dir_objective*). En Lituanien, le cas de l'objet direct est accusatif par défaut.

3. **verb_it** : verbes transitifs indirects. L'actant II en RSyntP se réalise comme objet indirect en RSyntS (*rel = indir_objective).* En Lituanien, l'objet indirect est au datif par défaut.
4. **verb_Ditr** : verbes ditransitifs. . L'actant III en RSyntP se réalise comme objet indirect en RSyntS (*rel = indir_objective).* En Lituanien, l'objet indirect est au datif par défaut.
5. **verb_Quad** : verbes avec quatre arguments

3. **Adjectifs** (déterminant, possessif, numéral, quantificateur),
4. **Adverbes** (préposition, conjonction, coordination, connecteurs, négation).

Les informations enregistrées dans *lexicon* sont récupérées par les règles de transduction (qui seront décrites dans le Chapitre 8). Elles sont suffisamment génériques pour s'appliquer à plusieurs langues. Afin de décrire le lituanien, nous n'avons eu à ajouter que deux paramètres dans cette partie du dictionnaire : l'ajout de cas par défaut du premier actant des noms et l'ajout de cas par défaut pour décrire le sujet, l'objet direct et l'objet indirect des verbes.

3. Lexèmes

La troisième partie de *lexicon* contient la liste des lexèmes. Nous y avons enregistré chaque lexème mentionné dans les valeurs du trait *lex* des entrées de *semanticon*. Chaque unité lexicale que nous avons enregistrée contient les informations suivantes : la catégorie grammaticale, la sous-catégorisation et les fonctions lexicales. Voici par exemple l'entrée de *nuosprendis* 'décision' dans *lexicon* :

```
nuosprendis : noun {
 lf = {
    name = Liqu1Func0
    value = keisti
  }

 lf = {
    name = Oper1
    value = priimti
  }
}
```

Le lexème *nuosprendis* est un nom. Il est la base de deux FL : $Liqu_1Func_0$ et $Oper_1$. Les collocatifs (verbes supports) du *nuosprendis*, *keisti* (‘changer’) *et priimti* (‘prendre’) sont enregistrés dans *lexicon* comme des verbes du type verbe_dt.

3.1 Traitement de locutions

Les locutions sont des lexèmes et font donc partie du *lexicon*. Toutefois, nous avons dû incorporer quelques changements dans le dictionnaire afin qu’elles puissent être arborisées par les règles de grammaire dans la prochaine étape du traitement. Nous expliquons ici comment nous avons géré l’enregistrement des locutions dans le *lexicon*.

Nous avons observé dans le corpus deux catégories de locutions :

1. Locutions figées du point de vue flexionnel. En lituanien, elles se comportent comme un « tout », c’est-à-dire qu’aucun composant de la locution n’est jamais fléchi.
2. Locutions fléchies. En lituanien, ces locutions sont fléchies « de l’intérieur », autrement dit elles possèdent un composant fléchi alors que les autres composants ne changent pas morphologiquement, et il est nécessaire d’accéder à sa structure syntaxique interne pour faire la flexion.

Les locutions adverbiales que nous avons présentées dans la section 4.2 tombent dans la première catégorie. Dans le *lexicon*, elles sont donc enregistrées entre guillemets suivies de la catégorie grammaticale, comme un seul bloc. Elles resteront inchangées lors du passage de RSém <–> RsyntP et RSyntP <–> RSyntS.

Les locutions nominales et verbales, elles, sont fléchies. Afin de les arboriser, nous avons besoin d’accéder à la racine de la locution (son composant fléchi). Dans les prochains paragraphes, nous expliquons les étapes du traitement pour ce type de locutions.

Nous avons classé les locutions fléchies selon la catégorie grammaticale de la racine (nom ou verbe) et le nombre de segments qui les composent. Les locutions nominales et verbales sont séparées en deux parties :

1. Nom ou verbe, composants principaux de la locution, qui définissent sa partie du discours
2. Le segment qui le complète ou le modifie

Prenons comme exemple la locution nominale *pataisos namai* (‘maison de correction’). Ici, *namai* (‘maison’) est le segment principal qui sera la racine du syntagme

nominal et qui portera la flexion. Le deuxième segment, *pataisa* (‘correction’) s’ajoutera dans l’arbre en tant que son actant, donc complément du nom, et dont la forme ne changera jamais.

La locution verbale *už grotų praleisti* (‘passer derrière les barreaux’) fonctionne de la même manière. Le verbe *praleisti* (‘passer’) sera le sommet du syntagme verbal et c’est lui qui portera la flexion. Le deuxième segment, *už grotų* (derrière les barreaux), sera un actant du verbe régi par la relation oblique. Nous n’avons pas besoin d’avoir accès à la structure syntaxique interne de cette partie de la locution puisqu’elle ne changera jamais morphologiquement et son ordre restera figé.

Les locutions qui se composent de trois segments auront deux actants. Par exemple, la locution nominale *Lietuvos apeliacinis teismas* (‘Cour d‘appel de Lituanie’) possède la racine *teismas* (‘cour’), qui porte la flexion, et deux autres actants : le modificateur *apeliacinis* et le complément *Lietuvos*. L’adjectif *apeliacinis* doit être fléchi puisqu’il s’accorde avec le nom *teismas*. Pour cette raison, nous ne pouvons pas le mettre dans un segment avec *Lietuvos*, qui, lui, ne change pas de forme, c’est pourquoi nous avons une locution à trois segments.

L‘organisation de la description des locutions dans le *lexicon* est illustrée dans les exemples ci-dessous. Dans le dictionnaire, les locutions sont enregistrées entre guillemets. Cela signifie que dans la RSyntP, elles apparaîtront en tant qu’unités inséparables. Elles seront décomposées en RSyntS à l’aide d’une règle de transduction spécifique. Celle-ci sera présentée dans le Chapitre 8. La catégorie grammaticale est déclarée comme pour les autres lexèmes du dictionnaire. La catégorie grammaticale englobe les traits de locution (idiom) : *type* de locution selon le nombre de segments, les valeurs des segments (w1, w2, etc.) et les types des relations (r1, r2) actancielles qu’elles occuperont en RSyntS.

```
"pataisos namai" : noun {
  idiom= {
    type = w1_w2
    w1 = namai
    r1 = noun_completive
    w2 = pataisa
  }
}
```

```
"Lietuvos apeliacinis teismas" : noun {
  idiom= {
    type = w1_w2_w3
    w1 = teismas
    r1 = modif
    w2 = apeliacinis
    r2 = noun_completive
    w3 = Lietuva
  }
}
```

De cette manière, nous avons décrit X lexèmes simples et X locutions. Plusieurs lexèmes possèdent des FL. Dans le Chapitre 4, nous avons brièvement décrit le fonctionnement des FL dans GÉCO. Regardons maintenant plus en détail comment les FL ont été modélisées sous des patrons génériques et quel système de règles a été appliqué afin de générer des collocations de plusieurs types. Après avoir décrit le fonctionnement du système, nous montrerons quelles FL nous avons introduites dans GÉCO afin de générer les collocations en lituanien.

Chapitre 7. Traitement des collocations du lituanien dans GÉCO

L'objectif du présent chapitre est d'expliquer comment nous avons traité les collocations du lituanien dans GÉCO. Premièrement, nous présentons la méthodologie de l'implémentation des collocations en reprenant les informations exposées dans (Lambrey & Lareau, 2015 ; Lambrey, 2016). Deuxièmement, nous donnons un inventaire des FL que nous avons modélisées à partir de corpus et à la fin, nous listons des FL que nous avons créées afin de générer des paraphrases.

1. Implémentation des collocations dans GÉCO

Au total, 26 259 fonctions lexicales sont implémentées dans GÉCO à l'aide de 129 règles de transduction (Lambrey, 2016). Dans cette section, nous nous intéressons aux techniques de l'implémentation des FL. Comme l'indique Lambrey (2016), elle a suivi une méthodologie en trois étapes : l'identification des régularités parmi les diverses collocations qui existent dans les langues, la modélisation de ces régularités et leur intégration dans GÉCO via des règles de transduction MATE. Nous allons maintenant parcourir chacune de ces étapes.

1.1 Identification des régularités

Dans le Chapitre 3, nous avons présenté la description des types de collocations qui existent dans la langue en général et comment elles sont modélisées dans le cadre de la théorie Sens-Texte. Le début du développement des FL dans GÉCO était basé sur l'inventaire des FL de MARQUIS. Dans le projet MARQUIS, il existait une trentaine de types de FL (cf. Figure 13).

Comme précisé dans (Lambrey, 2016) l'objectif de GÉCO était de continuer l'intégration de ces FL de manière exhaustive, en se focalisant sur les FL standard. Les configurations de FL ne sont pas traitées car il s'agit de combinaisons de plusieurs FL non reliées syntaxiquement, ce qui rend le mécanisme de formation difficile à modéliser. Le nombre de FL standard syntagmatiques, simples et complexes étant trop élevé (il n'existe pas de décompte), leur comportement a dû être généralisé : il a s'agît de les regrouper sous des patrons ou classes génériques (Lambrey, 2016 p. 83) :

> *« La création de patrons généraux, des classes de FL partageant des propriétés similaires, s'insère dans une perspective de développement continu de notre plateforme et vise à faciliter l'implémentation de nouvelles FL en plus du maintien des structures existantes. L'intérêt de créer des patrons est qu'ils font abstraction de la nature exacte des FL*

traitées et fonctionnent comme un modèle de lexicalisation plus général encore ».

Nous allons maintenant présenter la modélisation des FL.

$Func_0$	$IncepFunc_1$	$Labor_{23}$	$Oper_2$
$Func_1$	$IncepFunc_2$	$Labor_{31}$	$Oper_3$
$Func_2$	$IncepFunc_3$	$Labor_{32}$	$Magn.Oper_1$
$Func_3$	$Labor_{12}$	Magn	$Oper_{12}$
Adv_1	$Labor_{13}$	$Oper_0$	$Oper_{13}$
Adv_2	$Labor_{21}$	$Oper_1$	$Oper_{21}$
$Oper_{23}$	$Oper_{31}$	$Oper_{32}$	$Magn.Oper_2$

Figure 13 Liste des FL intégrées dans MARQUIS (Lambrey, 2016)

1.1 Modélisation des FL

La structure sémantique représente les sens et relations exprimés par la FL tandis que les noms des FL s'intègrent dans la représentation syntaxique profonde de l'énoncé. Pour illustrer le fonctionnement, nous tirons un exemple (cf. Figure 14) de Lambrey (2016). La figure décrit la réalisation des collocations *peur bleue*, *brouillard dense*, *heavy smoker* qui expriment un phénomène d'intensification (exemples de la FL Magn) en syntaxe profonde.

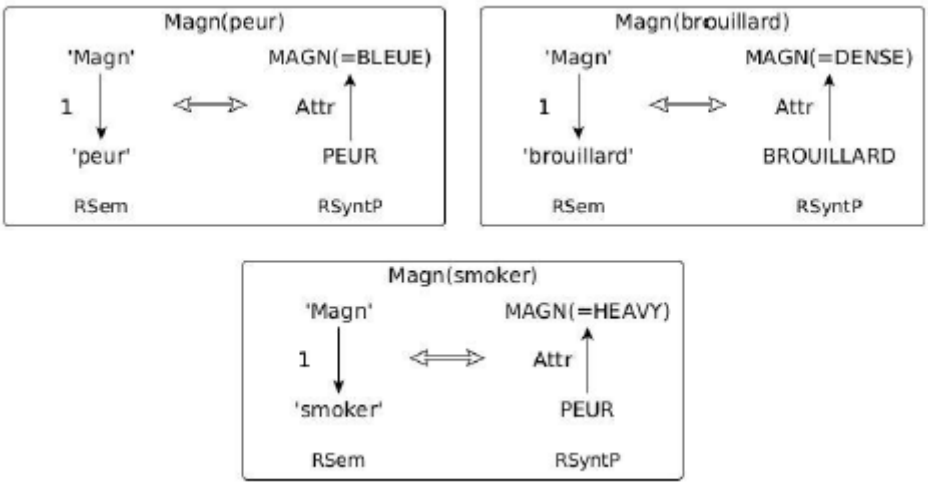


Figure 14 Généralisation du fonctionnement de Magn (Lambrey, 2016)

Comme il n'est pas possible de prévoir tous les sémantèmes possibles de Magn, le nom de FL Magn a été employé directement dans la RSém. La valeur de cette FL apparaîtra dans la RSyntP en tant que collocatif.

Plusieurs FL possèdent un comportement similaire à Magn, comme par exemple Ver, Bon, Pos, etc. dont le comportement peut aussi être généralisé sous le même patron de Modification.

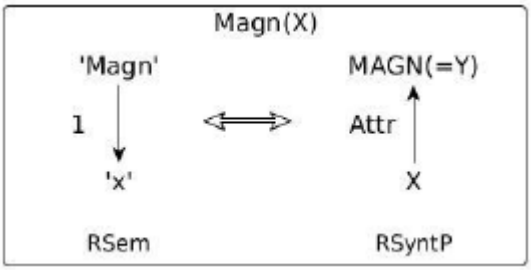


Figure 15 Modélisation générique de Magn (Lambrey, 2016)

De la même manière, une quarantaine de FL standard syntagmatiques simples ont été traitées en se basant sur les listes de FL standard courantes (Mel′čuk, 1996; Mel′čuk et al., 1995, 1999).

Les FL complexes ont été modélisées comme des compositions sémantiques à la Kahane & Polguère (2001). La Figure 16 tirée de (Lambrey, 2016) présente la modélisation en RSém des FL complexes AntiBon(choix) = *mauvais* ~ et $CausReal_1$(abîme) = *projeter dans* ART ~):

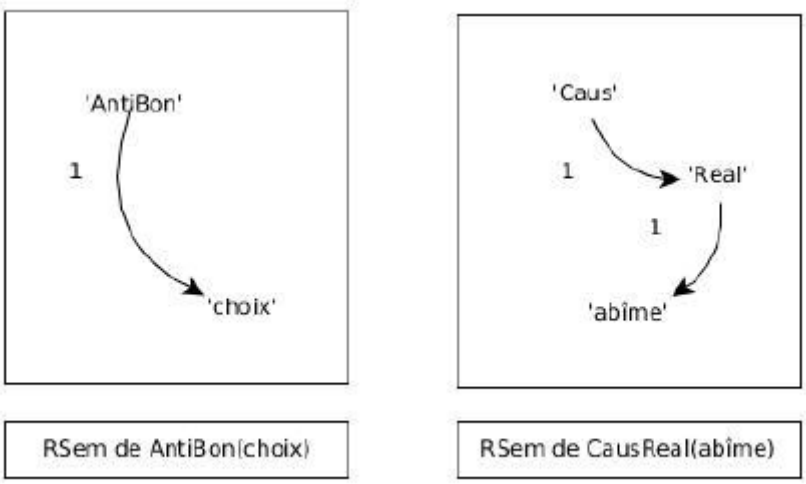


Figure 16 Représentation sémantique de FL complexes (Lambrey, 2016)

La formation des FL complexes s'applique aux patrons et non à chaque FL individuellement : « toutes les FL du patron Modification présentent les mêmes schémas de combinaison pour former des FL complexes. De ce fait, Magn, Ver et Pos se combinent de la même manière pour former AntiMagn, AntiVer et AntiPos. Il s'agit là d'un choix que nous avons fait pour faciliter le traitement des FL complexes » (Lambrey, 2016, p. 87). Enfin, il est à noter que GÉCO ne couvre pas les configurations des FL, car leurs configurations se modélisent différemment en RSém (elles ne peuvent pas être décrites à l'aide d'une composition sémantique).

1.2 Intégration dans MATE

Afin de lexicaliser les fonctions lexicales, GÉCO s'appuie sur les informations encodées dans trois dictionnaires : le *semanticon*, qui liste les lexicalisations possibles de chaque sémantème, le *lexicon* contenant toutes les entrées lexicales d'une langue et le *lf*, dictionnaire de fonctions lexicales qui inclut les informations sur les FL et leur fonctionnement en représentation syntaxique profonde. Regardons maintenant plus en détail comment ces aspects sont implémentés dans MATE.

1.2.1 Dictionnaires

Les trois dictionnaires, le *semanticon*, le *lexicon* et le *lf* ont été repris directement de MARQUIS. Cependant, afin de pouvoir intégrer de nouvelles FL dans le dictionnaire *lf*, l'organisation du dictionnaire a dû être repensé. Les changements ont porté sur l'intégration

de l'information sémantique : un attribut « sem » a été ajouté pour chaque FL ou chaque patron de FL. Lambrey (2016) l'illustre avec un extrait du dictionnaire *lf* suivant :

```
lf{

verb {
        dpos = V
  gp = {
                I = { dpos = N }
                II = { dpos = N }
                II = { dpos = N }
  }
}

// Verbes supports
Oper : verb {
  gp = { L = II }
}
Oper0 : Oper {}

// Verbes de réalisation
Real0 : Oper0 { sem=Real }
```

Cet extrait définit une classe « verb » de FL. Cette classe contient les attributs « dpos » (partie du discours profonde) et « gp » (patron de régime).

La FL Oper est une sous-classe de classe « verb ». Son gp indique que la base (L) se réalise comme l'actant syntaxique profond II.

Dans cet extrait, nous pouvons également voir deux regroupements de FL : verbes supports et verbes de réalisation. Sem = Real signifie que la FL $Real_0$ est sémantiquement pleine. La FL Real est une sous-classe de Oper puisqu'elle possède les mêmes propriétés syntaxiques que $Oper_0$. Malgré le fait que Real n'est pas un sous-type de Oper d'un point de vue sémantique, cette architecture, selon Lambrey (2016) évite la redondance dans la description des FL : la seule différence existante entre ces deux FL est que $Real_0$ est sémantiquement pleine : (sem=Real). Pour conclure, Le dictionnaire *lf* regroupe les propriétés sémantiques et les combinatoires des FL.

Dans le *lexicon*, pour chaque entrée d'unité lexicale, un nouveau nœud « lf » avec les attributs « name » et « value » a été inséré. Cette conception est différente de MARQUIS et elle facilite la récupération des informations lors de l'application des règles de transduction. Voici un extrait simplifié du *lexicon* en français :

```
fumée : noun {
  lf = {
    name = Figur
    value = rideau
  }
}
```

Comme nous pouvons le voir, la représentation et le fonctionnement des FL ont été modélisés sous forme de règles de correspondance entre le niveau sémantique et le niveau syntaxique profond. Le passage s'effectue à l'aide des règles de transduction qui forment la grammaire de GÉCO. Afin de se familiariser avec les règles de transduction implémentées dans GÉCO, le lecteur est invité à se référer à (Lambrey, 2016 pp. 91-94).

1.2.2 Inventaire des patrons

Dans GÉCO, il existe 7 patrons de FL : « support_verbs », « semantic_verbs », « semantic_governon_noun », « realisation_verbs », « preposition, no_patterns », « modification », « adjectival_complement ». Chaque patron possède un répertoire de règles de transduction. Nous allons maintenant décrire brièvement les patrons Modification, Verbes Supports, Verbes de réalisation et Combinaisons des FL verbales.

Modification

Les FL Magn, Bon, Ver et Pos renvoient à un sens d'intensification ou de louange. Ces FL sont sémantiquement pleines et leur nom doit faire partie de la structure sémantique de la phrase. Dans la RSém, la position actancielle 1 est toujours occupée par la base. Dans la RSyntP, cette relation se réalise par la relation ATTR entre la base et le collocatif.

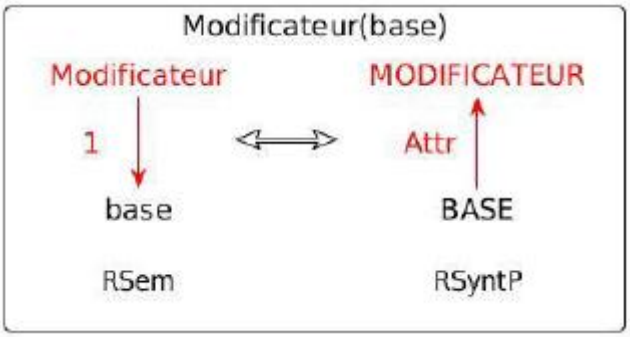


Figure 17 Fonctionnement du patron Modification (Lambrey, 2016)

Préposition

Certaines FL, telles que Loc_{ad}, $Loac_{ab}$, Instr, etc. introduisent une préposition dans la RSyntP de la phrase. Ces FL sont prédicatives et possèdent deux actants en RSém. Le premier instancie une action ou une personne, tandis que le deuxième (la base) l'entité dans laquelle se passe l'action (Loc_{in}) ou l'entité cible de l'action (Loc_{ad}), etc. Dans la RSyntP, la relation est inversée entre le collocatif et l'actant 1 (donc relation ATTR). L'actant 2 de RSém se réalise par la relation II. Ces FL sont modélisées sous le patron Préposition. Le fonctionnement de ce patron est illustré dans la Figure 18.

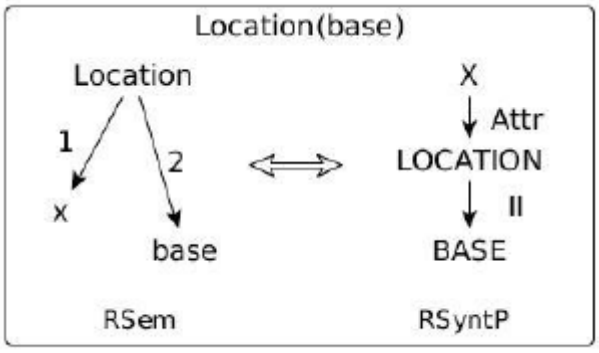


Figure 18 Fonctionnement du patron Location (Lambrey, 2016)

Verbes supports

Un verbe support est un verbe sémantiquement vide, il ne fait pas partie de la RSém, mais apparait dans la RSyntP. La théorie Sens-Texte distingue trois types de verbes support : Func, Oper et Labor, selon la position de la base par rapport au verbe. Les indices de ces FL, par exemple $Oper_{1,}$ représentent la position syntaxique des actants de la base par rapport au verbe. Dans l'implémentation de Lambrey (2016), ces indices font référence aux actants sémantiques de la base, même si dans la littérature on fait plutôt référence aux actants syntaxiques. Dans GÉCO, il y a trois règles de transduction de type Verbes Supports : « lex_vsupp_0 » ($Func_0$ et $Oper_0$), « lex_vsupp_i » ($Oper_i$, $Func_i$,) et « lex_vsupp_ij » ($Oper_{ij}$, $Func_{ij}$, $Labor_{ij}$).

Verbes de réalisation

Les verbes de réalisation, Real, Fact, et Labreal sont sémantiquement pleins, mais ils fonctionnent de la même façon que les verbes support dans la RSyntP. De ce fait, il existe trois règles dont le fonctionnement est similaire à celles des verbes support : « lex_vreal_0 »,

« lex_vreal_i » et « lex_vreal_ij ». Ces règles lexicalisent le verbe de réalisation, la base et les actants selon l'indice des FL. La différence entre les règles de verbes support et verbes de réalisation est que celles dernières portent l'étiquette « sem » : le sens des verbes de réalisation apparait dans la RSém de la phrase.

Combinaisons des FL verbales

Les FL phasiques (Incep, Fin, Cont) n'ont pas de structure actancielle et doivent s'appuyer sur les FL actancielles. Elles qualifient un « état ou événement » et ne peuvent s'appliquer que sur un seul actant : une FL verbale. Trois règles de transduction ont été créées en fonction du nombre d'actants des FL verbales. Les FL phasiques sont encodées à l'aide de compositions sémantiques, le nœud des FL phasiques fait partie de la RSem, comme illustré dans la Figure 19.

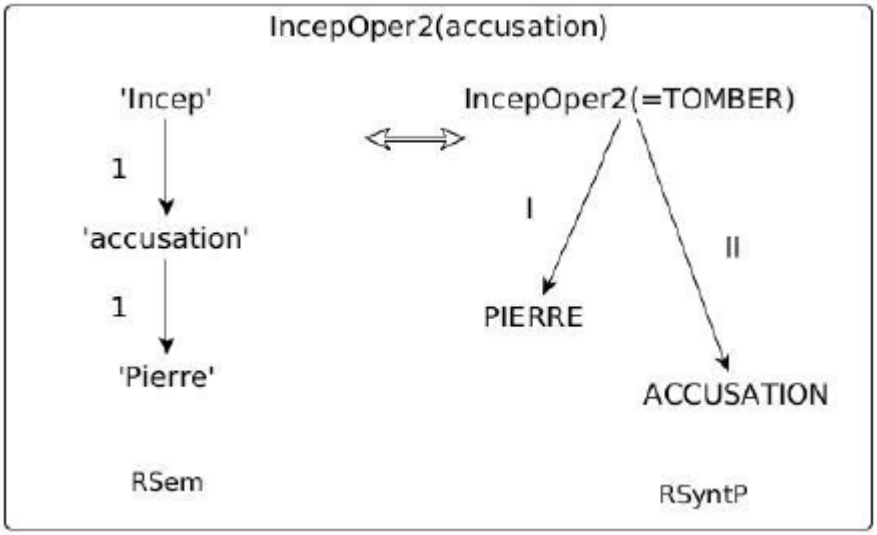


Figure 19 Fonctionnement de IncepOper2(accusation) (Lambrey, 2016)

Les trois FL causatives (Caus, Liqu, Perm) introduisent un nouvel élément dans la structure actantielle (l'agent). Celui-ci se réalise par la relation I sur le collocatif verbal et décale les autres actants du verbe.

Les combinaisons des FL phasiques et causatives sont générées à l'aide des règles génériques de combinaison FL causatives + phasiques + verbe support, qui prévoient toutes les combinaisons possibles : CausIncepOper, PermFinFunc, etc.

GÉCO offre une grande combinatoire des FL. Voici un exemple tiré de (Lambrey, 2016) :

- (Incep|Fin|Cont|Prepar|Prox|Non) (Vsupp|Vreal|Vsem)

- (Caus|Liqu|Perm) (Vsupp|Vreal|Vsem)
- (Caus|Liqu|Perm)$_i$ (Vsupp|Vreal|Vsem)
- (Caus|Liqu|Perm) (Incep|Fin|Cont|Prepar|Prox|Non) (Vsupp|Vreal|Vsem)
- (Caus|Liqu|Perm)$_i$ (Incep|Fin|Cont|Prepar|Prox|Non) (Vsupp|Vreal|Vsem)

Nous venons de décrire quelques patrons génériques de FL dans GÉCO. Nous avons dû en prendre connaissance pour pouvoir modéliser et générer de diverses collocations en lituanien. L'implémentation des collocations du lituanien fait l'objet de la prochaine section.

2. Implémentation des collocations du lituanien dans GÉCO

Afin de constituer notre corpus, nous avons parcouru plusieurs articles de nouvelles judiciaires. Nous avons remarqué que le type et le nombre de collocations est souvent similaire par rapport à la longueur de l'article (moyenne). Dans notre article, nous avons trouvé au total 15 collocations et nous les avons modélisées via les fonctions lexicales.

Nous avons identifié six FL de type **verbe support** :

	Lituanien		Traduction en français	
FL	**Base**	**Collocatif**	**Base**	**Collocatif**
Oper$_1$	išvada	*padaryti*	conclusion	*faire*
Oper$_1$	nusikaltimas	*įvykdyti*	crime	*commettre*
Oper$_1$	žala	*atlyginti*	dommage	*compenser*
Oper$_2$	grasinimas	*sulaukti*	menace	*reçevoir*
Func$_1$	pavydas	*apimti*	jalousie	*envahir*

Deux FL de type **modification** :

	Lituanien		Traduction en français	
FL	**Base**	**Collocatif**	**Base**	**Collocatif**
Magn	grasinimas	*baisus*	menace	*horrible*

Magn	nusikaltimas	*kraupus*	crime	*terrifiant*

Trois FL de type **verbes de réalisation** :

	Lituanien		Traduction en français	
FL	**Base**	**Collocatif**	**Base**	**Collocatif**
Real$_0$	byla	*nagrinėti*	cas	*examiner*
Real$_1$	grasinimas	*įvykdyti*	menace	*effectuer*
Real$_2$	prašymas	*tenkinti*	demande	*approuver*

Cinq FL **complexes** :

	Lituanien		Traduction en français	
FL	**Base**	**Collocatif**	**Base**	**Collocatif**
AntiReal$_2$	Išvada	*padaryti*	conclusion	*faire*
Liqu$_1$Func$_0$	nuosprendis	*keisti*	verdict	*changer*
IncepFact$_0$	Nutartis	*įsiteisinti*	verdict	*devenir valide*
Liqu$_1$Func$_0$	Ryšys	*nutraukti*	relation	*rompre*
LiquReal$_2$	Namai	*išprašyti*	maison	*rejetter*

Une FL **nominale quantitative**:

	Lituanien		Traduction en français	
FL	**Base**	**Collocatif**	**Base**	**Collocatif**
Mult	teisėjas	*kolegija*	juge	*collège*

Le nombre de collocations dans le corpus n'est pas très élevé mais on y trouve pourtant une grande variété de FL : FL de type Magn, FL verbales, FL nominales, FL complexes, etc. Notre corpus satisfait donc notre objectif de tester la performance de GÉCO : le système devra générer des collocations de plusieurs types.

Afin d'intégrer les collocations dans GÉCO, nous avons dû comprendre l'architecture de la modélisation des FL de (Lambrey, 2016) que nous venons de décrire dans la section précédente. Comme nous l'avons expliqué dans la section 1.3.2, les FL Magn, Mult, Real, Loc_{in} ainsi que les FL phasiques et causatives apparaissent dans la RSém de la phrase. Les phrases 2, 3, 4, 5, 9, 10, 11, 13, 14, 16. comportent le composant sémantique des FL. Seules les FL verbales, Oper et Func, ne font pas partie des structures sémantiques, les collocatifs étant sémantiquement vides et donc intégrés directement dans le *lexicon*.

3. Génération de paraphrases

Un des objectifs principaux de la GAT est de construire un modèle linguistique exhaustif et puissant. Selon Iordanskaja, Kittredge & Polguère (1991), une mesure de la puissance et de l'exhaustivité d'un modèle de langue est sa capacité à représenter toutes les façons possibles qu'un locuteur humain pourrait choisir de « dire la même chose » en utilisant la connaissance linguistique (par opposition à la connaissance du monde). Selon Milićević (2007), décrire une langue, « c'est avant tout modéliser la capacité des locuteurs de reconnaitre et de produire les paraphrases ». Comme nous l'avons vu dans la section 1.2 du Chapitre 4, la génération de paraphrases est au cœur de la Théorie Sens-Texte. Mel'čuk (1992) écrit « une des tâches primordiales de la linguistique théorique contemporaine est l'élaboration d'une théorie de la paraphrase langagière ».

Dans le cadre de la GAT, la paraphrase a pour objectif d'augmenter la diversité des textes générés. La paraphrase peut être de nature lexicale (synonymie) ou structurale (différentes constructions syntaxiques). Dans une tentative de résoudre le problème algorithmique de génération de paraphrases, plusieurs travaux de recherche ont porté récemment sur l'identification et la génération de paraphrases : Barzilay & McKeown, 2001 ; Barzilay & Lee, 2003.

La Théorie Sens-Texte est un des premiers formalismes à modéliser la génération de paraphrases ; elle vise à produire toutes les RSyntP possibles à partir d'une seule RSém. D'après la théorie, la modélisation de la capacité paraphrastique repose sur la conception des règles linguistiques capables de décrire des **liens** paraphrastiques. Ces liens paraphrastiques

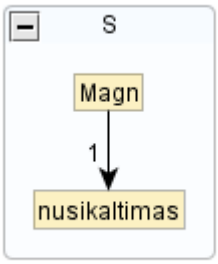
sont universels : « un lien paraphrastique donné peut tout aussi bien lier des expressions appartenant à une seule langue que celles appartenant à des langues différentes. Cela veut dire que la paraphrase peut être approchée non seulement dans une perspective intralinguistique, mais aussi dans une perspective interlinguistique et qu'elle peut être modélisée par des règles linguistiques universelles » (Milićević, 2007). Ces règles universelles de paraphrasage, qui « reposent sur le recours aux FL, permettent d'obtenir automatiquement toutes les paraphrases similaires dans toute langue – pourvu, bien entendu, qu'on dispose, pour cette langue, d'un dictionnaire où les FL (autrement dit, les collocations) sont spécifiées pour chaque lexie vedette » (Mel'čuk, 2013, p. 137).

La modélisation linguistique de GÉCO permet deux types de paraphasage : **paraphrase lexicale** et **paraphrase lexico-syntaxique**. La paraphrase lexicale se produit à l'aide des liens synonymiques entre les mots.

Dans le *semanticon*, nous avons fourni plusieurs lexicalisations possibles pour chaque sémantème donné. Prenons un exemple d'un adverbe *nuolat* ('constamment') qui peut se lexicaliser de quatre façons différentes : deux autres adverbes synonymes (*nepaliaujamai* et *vis*) et deux locutions adverbiales *be atvangos* et *be paliovos*.

```
nuolat {
        lex = nuolat
        lex = "be atvangos"
        lex = "be paliovos"
        lex = nepaliaujamai
        lex = vis
}
```

Autrement dit, la génération de paraphrase lexicale de *nuolat* repose sur le répertoire des synonymes fourni dans le *semanticon*, ce qui inclut les liens de « synonymie » entre lexèmes de parties du discours différentes (comme *décider~décision*). Les synonymes sont également fournis pour les valeurs des collocatifs dans le *lexicon*. Prenons, par exemple une structure sémantique suivante :



Dans le lexicon, deux valeurs de Magn sont spécifiées :

```
nusikaltimas : noun {
  lf = {
    name = Magn
    value = kraupus
  }
  lf = {
    name = Magn
    value = baisus
  }
```

Ainsi, à partir d'une seule représentation sémantique nous sommes capables d'obtenir deux paraphrases pour *nusikaltimas* ('crime') : *baisus* et *kraupus* (synonymes d'horrible).

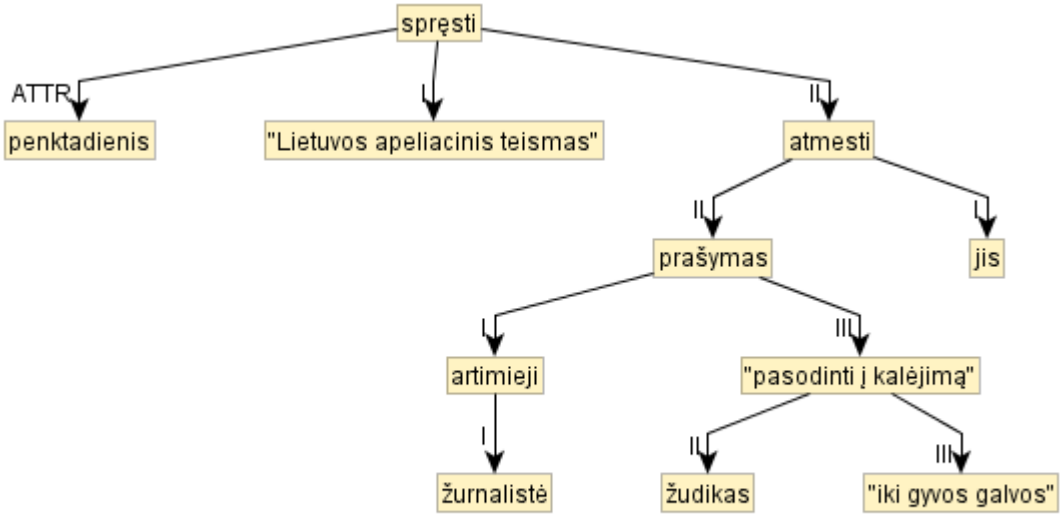
La deuxième façon de générer des paraphrases dans GÉCO repose sur la génération de **paraphrase lexico-syntaxique** : plusieurs structures syntaxiques sont générés à partir d'une seule représentation sémantique. Prenons, par exemple l'entrée de *nutarimas* ('décision') qui est une des lexicalisations possibles de sémantème 'spręsti' ('décider').

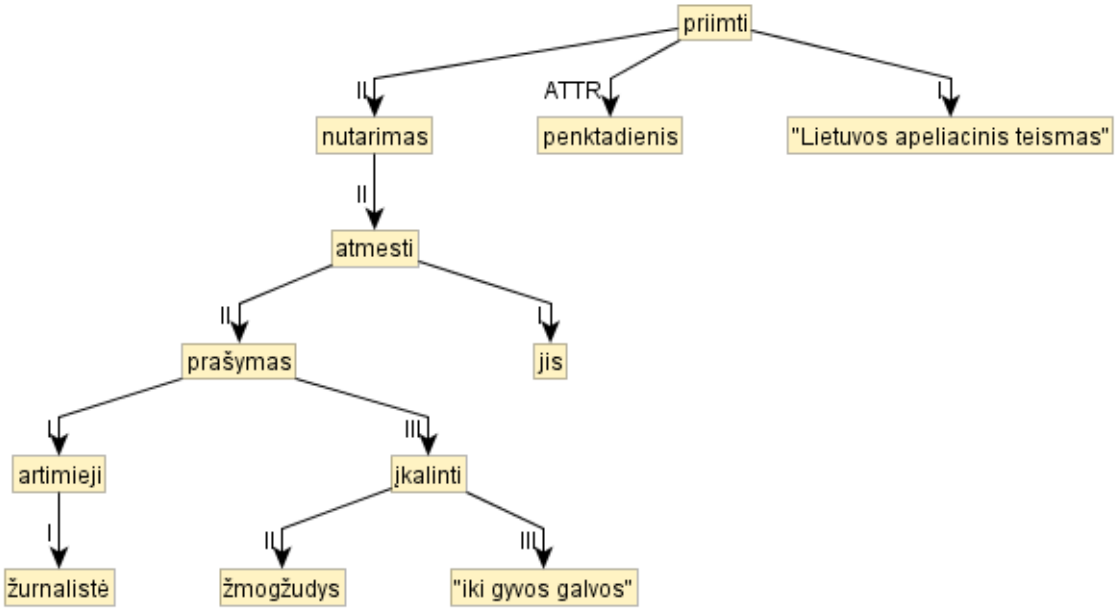
```
nutarimas : noun {
  lf = {
    name = IncepFact0
    value = įsiteisinti
  }
   lf = {
    name = Oper1
    value = priimti
  }
}
```

Ainsi, pour la Phrase 2, où le sémantème 'spręsti' est utilisé, nous obtenons deux structures syntaxiques différentes :

1. *Lietuvos apeliacinis teismas sprendžia* ‘La Cour d’appel de Lituanie décide’
2. *Lietuvos apeliacinis teismas priima sprendimą* ‘La Cour d’appel de Lituanie prend une décision’

Les arbres syntaxiques générés en RSyntP sont illustrées ci-dessous :





La première phrase est réalisée par le verbe *spręsti*, tandis que la deuxième emploie l'utilisation de verbe de support *priimti*, qui est fourni dans dans la valeur de FL $Oper_1$ dans l'entrée de *nutarimas*. Le sens de la phrase est le même mais les structures syntaxiques générées sont différentes.

Grâce à ces techniques, nous avons obtenu plusieurs paraphrases pour presque chaque structure de départ. Voici le nombre de paraphrases produites par phrase du corpus :

Phrase	**Nombre de paraphrases générées**
Phrase 1	2
Phrase 2	32
Phrase 3	18
Phrase 4	4
Phrase 5	2
Phrase 6	2
Phrase 7	5
Phrase 8	0
Phrase 9	4
Phrase 10	6
Phrase 11	4
Phrase 12	4
Phrase 13	4
Phrase 14	35
Phrase 15	0
Phrase 16	0

Chapitre 8. Grammaire de GÉCO

La grammaire de GÉCO est une grammaire de graphes. Ce formalisme permet de représenter un large éventail de structures linguistiques, telles que structures sémantiques, arbres de dépendance, structures morphologiques, etc. Les graphes utilisés pour représenter les structures linguistiques se composent de nœuds, d'attributs et d'arcs. Les nœuds sont connectés par des arcs étiquetés. Le transfert de graphes linguistiques est effectué par une grammaire transductive. Elle est appelée transductive car son principe est de construire, à partir d'un graphe donné, un graphe correspondant de niveau adjacent, de façon non destructive. Le processus de transfert est piloté par des règles qui constituent la grammaire de GÉCO.

Dans le Chapitre 4 de l'état de l'art, nous avons illustré le fonctionnement de la règle de lexicalisation simple dans GÉCO. Nous avons appris que les règles de grammaire de GÉCO sont des règles de correspondance avec un **côté gauche**, un **côté droit**, un ensemble de **conditions d'application** qui doivent être tenues pour que la règle soit applicable. Le **côté gauche** signifie qu'une règle peut partager un fragment d'entrée avec d'autres règles. Le **côté droit** contient un fragment cible qui doit déjà être créé par d'autres règles.

Si un fragment de graphe source correspond au côté gauche d'une règle et si les conditions spécifiées sont remplies, le fragment est mappé sur le sous-graphe spécifié sur le côté droit. Cela signifie que, pour interpréter les règles, le mapping d'un graphe source vers un graphe cible est effectué itérativement. Chaque itération se compose de cinq étapes qui sont appliquées à toutes les règles en parallèle : correspondance, évaluation, regroupement, application et unification.

Nous allons maintenant parcourir les modules de règles de GÉCO. Nous avons pris les fragments des modules **RSém-RsyntP** et **RSyntP-RSyntS** de MARQUIS. Les modules contiennent les règles de base, telles que lexicalisation et attribution des actants syntaxiques. Afin de générer nos phrases, nous avons dû créer de nouvelles règles. Elles seront décrites dans la section 2.

1. Modules de règles

Dans GÉCO, il existe pour l'instant deux modules de règles de grammaire : « module 20 » (RSem - RSyntP) et « module 25 » (RSyntP-RSyntS). Dans chaque module, les règles

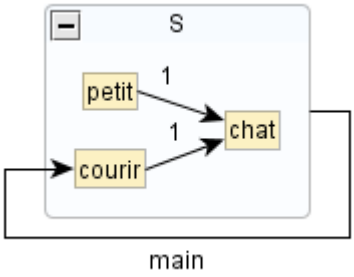
sont regroupées dans les paquets. Regardons maintenant de plus près quelles règles existent dans chacun de ces modules.

1.1 RSem à RSyntP

La grammaire de **RSem-RSyntP** fonctionne via un algorithme top-down. En d'autres termes, la création du graphe de sortie de RSyntP commence à sa racine supérieure. Ensuite, les arcs et les nœuds sont créés jusqu'à ce que tous les nœuds du graphe RSém soient consommés par les règles. Nous allons maintenant présenter quelques règles de base qui permettent de générer la majeure partie du graphe de sortie.

1.1.1 Règle synt_root : construction de la racine de l'arbre

Dans GÉCO, la structure communicative est encapsulée dans la structure S de la structure sémantique. Cela permet de définir le nœud le plus saillant communicativement. Le graphe sémantique tiré de GÉCO représentant la phrase *Le petit chat court* est représenté à la figure ci-dessous.



Dans ce cas-ci, le nœud le plus saillant communicativement est ‘courir’, ce la relation « main ». Si nous mettions ‘chat’ comme nœud le plus saillant, la phrase produite serait *Le petit chat qui court*, et si nous choisissions ‘petit’, cela donnerait *Le chat qui dort est petit*.

La règle **synt_root**, exploite donc la structure communicative du graphe d'entrée en créant une correspondance entre le nœud sémantique le plus saillant et la racine du graphe de sortie. Cette règle crée ainsi le premier nœud de l'arbre de sortie. Le nœud n'est pas encore lexicalisé. La lexicalisation sera effectuée grâce aux règles de lexicalisation.

1.1.2 Règles de lexicalisation

Il existe quatre règles de lexicalisation dans GÉCO : **lex_standard**, **lex_special**, **lex_check_dpos_gp** et **lex_check_dpos_guess**.

Une fois qu'un nœud existe dans l'arbre de sortie, il doit être lexicalisé avant de pouvoir créer ses actants. La règle de **lexicalisation simple** (lex_standard) lexicalise les noms, les verbes, les adverbes et les adjectifs. Cette règle dépend fortement du *semanticon* et du *lexicon*. Elle permet de trouver les correspondants des nœuds des structures sémantiques (sémantèmes enregistrés dans le *semanticon*) dans le *lexicon*. Lorsque plusieurs lexicalisations existent, GÉCO crée autant de graphes de sortie qu'il y a de lexicalisations données dans le *semanticon*. Ainsi, la génération au sein de GÉCO est non déterministe.

Les unités, telles qu'heure, date, etc., sont lexicalisées séparément, par la **lexicalisation spéciale** (lex_special).

Les deux dernières règles, **lex_check_dpos_gp** et **lex_check_dpos_guess** sont appliquées lorsque GÉCO doit devenir la lexicalisation (le *dpos* n'a pas été enregistré). Maintenant que le nœud est lexicalisé, ses actants syntaxiques peuvent être crées. C'est la tâche des règles d'arborisation.

1.1.3 Règles d'arborisation

Il existe six règles d'arborisation dans GÉCO : **synt_actant**, **synt_actant_guess**, **synt_attr_Num**, **synt_attr_class**,

Une fois qu'un nœud est lexicalisé, ses actants syntaxiques peuvent être créés. La règle **synt_actant** permet de créer les dépendants d'un nœud qui est déjà lexicalisé en s'appuyant uniquement sur les informations du *lexicon*. La règle va chercher les informations dans le *gp* du lexème, récupérant ainsi les informations sur la réalisation syntaxique de ses actants.

Les trois autres règles sont expérimentales et ne servent pas dans notre implémentation ; nous les laissons donc de côté dans notre exposé.

1.2 RSyntP- RSyntS

L'application des règles commence à partir de la racine de l'arbre RSyntP jusqu'à ce que tous les nœuds d'entrée soient consommés. Les nœuds de l'arbre RSyntS sont étiquetés avec des lexèmes complets et les arcs avec des relations syntaxiques de surface telles que sujet, objet direct, objet indirect, etc. Les règles de ce module gèrent l'insertion des mots grammaticaux et l'arborisation des locutions.

Les règles de lexicalisation de surface sont : **lex_lu**, **lex_special** et **lex_det**. Les deux premières règles recopient les lexicalisations des lexèmes choisis en RSyntP en leur

attribuant les catégories syntaxiques de surface (*spos*). La troisième règle attribue des déterminants définis ou indéfinis aux noms, selon le trait de définitude enregistré dans les structures sémantiques.

Il existe trois règles de réalisation des actants : **synt_actant_dir**, **synt_subj_dir**, **synt_actant_prep**. Les deux premières règles réalisent les relations syntaxiques profondes (I, II, III) selon le régime du lexème qui les gouverne (il y a une règle spéciale pour le sujet, qui ne doit s'appliquer qu'avec un verbe fini). La troisième règle s'applique s'il y a des prépositions spécifiées dans le régime.

Ces règles, héritées de MARQUIS, ne suffisent pas pour couvrir tous les phénomènes linguistiques de notre corpus. Afin de générer les arbres syntaxiques de référence, nous avons dû créer de nouvelles règles dans les deux modules. Nous les présentons dans la prochaine section.

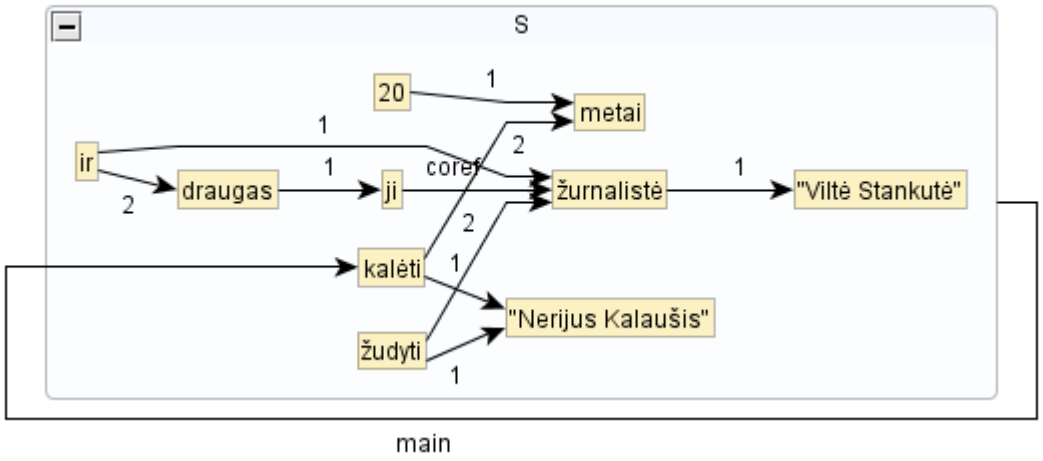
2. Nouvelles règles créées dans GÉCO

Nous avons dû ajouter à GÉCO quatre règles du module RSém-RSyntP et deux règles du module RSyntP-RSyntS. Nous allons maintenant présenter la modélisation de ces règles et leur fonctionnement dans MATE.

2.1 Module RSyntP

2.1.1 Gestion des pronoms et adjectifs possessifs

Une fois qu'une règle est appliquée à un nœud, celui-ci est consommé et ne peut plus être consommé par d'autres règles. Cela complique l'arborisation des structures réentrantes. Afin de résoudre ce problème, nous avons ajouté les anaphores dans les structures sémantiques. Les nœuds sont dupliqués pour gérer les pronoms comme 'il', 'elle', etc. Ainsi, pour traiter la Phrase 1, où le même nœud *žurnalistė* ('journaliste') doit être utilisé comme actant de *žudyti* ('tuer') et *draugas* ('ami'), nous avons ajouté un pronom *ji* ('elle') qui crée une coréférence sémantique entre *draugas* et *žurnalistė*. Voici un exemple graphique de la phrase qui comporte une relation *coref* entre deux nœuds :

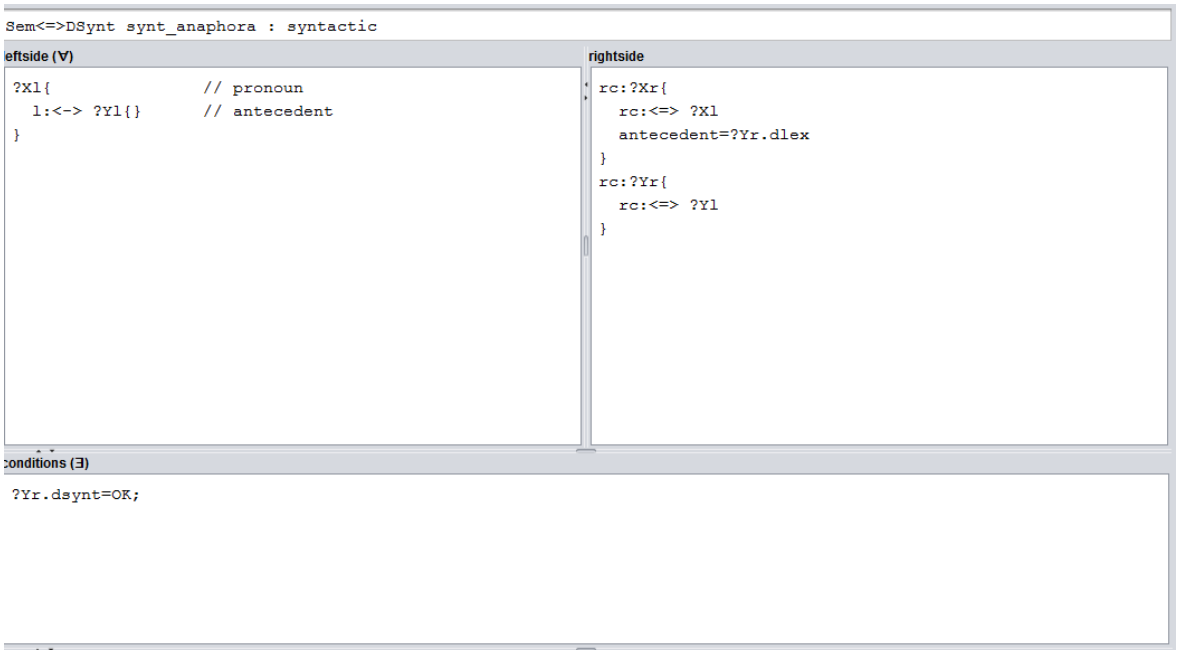


Voici la structure sémantique sous forme textuelle tirée de MATE :

```
ji:9{
  sem=ji
  <-> žurnalistė:5
}
```

Ici, <–> fait la référence entre les sémantèmes *ji* et *žurnalistė*.

Après avoir modélisé les anaphores dans les structures sémantiques, nous avons créé une nouvelle règle de grammaire permettant de récupérer le pronom et son antécédent. Ainsi, la même règle pourra être appliquée deux fois et permettra l'arborisation correcte lors du passage vers RSyntP. Voici la règle que nous avons créée sur MATE :



Côté gauche

Deux nœuds, *?Xl* et *?Yl* sont spécifiés. Le nœud *?Xl* instancie le pronom et *?Yl* instancie son antécédant. Les deux nœuds sont reliés par la relation <−> qui, comme nous avons vu, est spécifiée dans la structure sémantique de la phrase. Cette relation est consommée par la règle (*l :*).

Côté droit

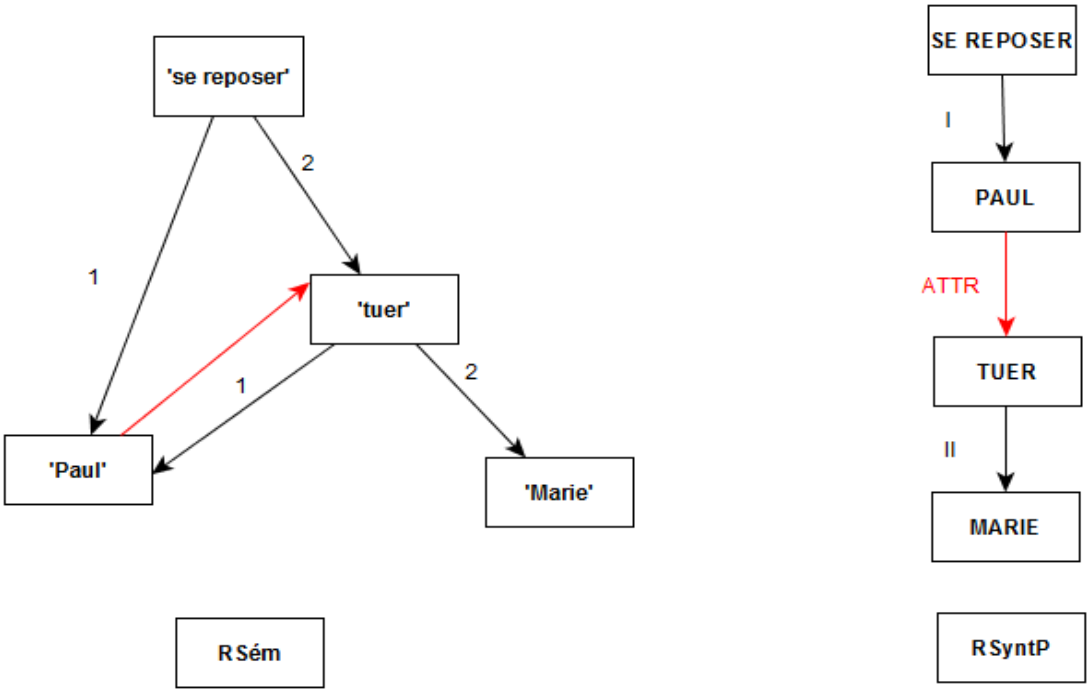
Le contexte de droit est défini (*rc :*). Il s'instancie sur le nœud *?Xr* qui existe déjà dans le graphe de sortie. La règle crée une relation entre *?Xr* et *?Xl*, le nœud *?Xr* devient donc le correspondant de *?Xl*. La règle crée un attribut *antecedent* qui prend pour valeur le *?Yr.dlex*. Celui-ci, comme défini par la règle plus bas, existe déjà dans le graphe de sortie (*rc :?Yr*) et il est correspondant de *?Yl*.

Conditions d'application

Les conditions de la règle attendent que le nœud *?Yr* soit lexicalisé pour s'appliquer (*?Yr.dsynt=OK*).

2.1.2 Phrases relatives

Pour générer des relatives du type *Paul, ayant tué Marie, se repose*, nous les avons représentées dans MATE comme à la figure ci-dessous. Quand MATE applique les règles, le relation 1 entre 'tuer' et 'Paul' ne se réalise pas conformément au régime de TUER, mais par une relation inversée ATTR, c'est-à-dire que PAUL gouverne TUER par la relation ATTR. Elle peut être illustrée comme dans la figure ci-dessous :



Voici le fonctionnement de la règle dans MATE.

```
Sem<=>DSynt synt_relative_lex : syntactic
leftside (∀)
l:?Xl{          // main predicate of the relative
  l:?r-> ?Yl{}  // noun to be modified
}
?L <- semanticon::(?Xl.sem).(lex)
rightside
rc:?Yr{         // noun to modify
  rc:<=> ?Yl
  ATTR-> ?Xr{   // main predicate of the relative
      <=> ?Xl
      dlex=?L
      dpos=lexicon::(?L).(dpos)
      dsynt=OK
  }
}
conditions (∃)
?Yr.dsynt=OK;
semanticon::(?Xl.sem).(lex);
lexicon::(?L).(dpos)=V;
not ?r=coref;
```

Côté gauche

La règle instancie les noeuds *?Xl*, qui fait référence au prédicat de la relative et *?Yl*, qui réfère au nom à modifer. En plus des nœuds, une variable *?L* est déclarée. Grâce à l'instruction *semanticon::(?Xl.sem).(lex)*, elle prend pour valeur les lexicalisations du sémantème instancié au nœud *?Xl*.

Côté droit

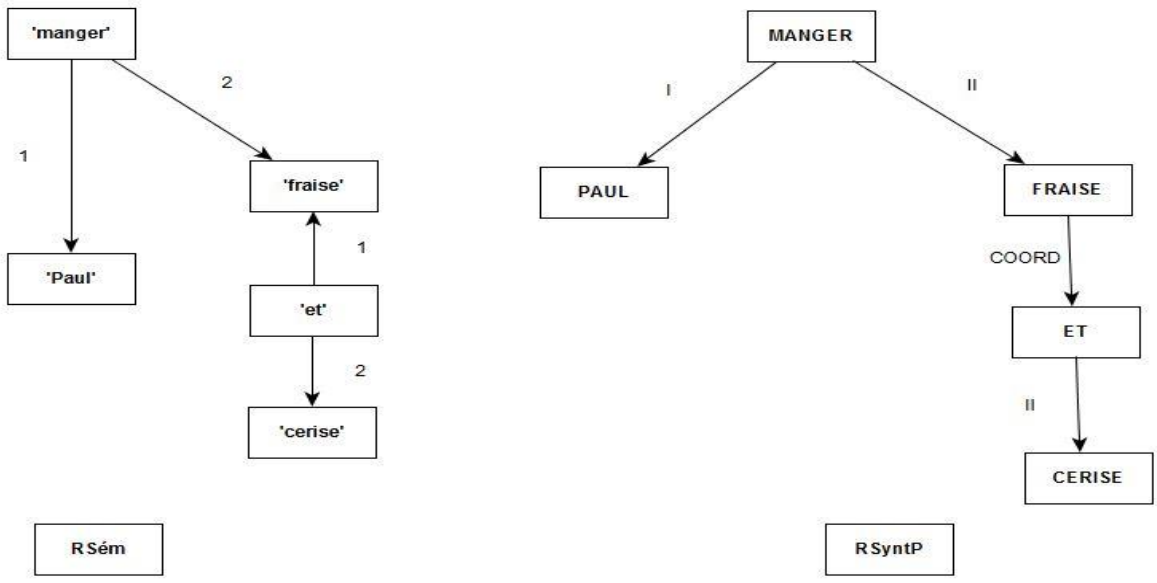
Le noeud *?Yr* existe déjà dans le graphe de sortie et il est le correspondant de *?Yl* à gauche, ce qui est marqué par l'instruction *rc :?Yr ⇔?Yl*. Le noeud contenant le nom est donc déjà créé dans le graphe de sortie. La règle crée un noeud *?Xr*, correspondant au *?Xl* du côté gauche (prédicat de la relative). La relation entre les noeuds *?Yr* et *?Xr* est spécifiée par la relation ATTR en RSyntP. Un attribut dlex est créé au noeud *?Xr*. Il correspond à la valeur de la variable *?L*. Sa catégorie grammaticale en RSyntP (dpos) est récuperée par la commande *lexicon::(?L).(dpos)*. Cette commande accède à l'entrée du lexème *?L* dans *lexicon* et reprend la valeur de son trait *dpos*. Ce noeud est ensuite marqué comme lexicalisé (*dsynt=ok*).

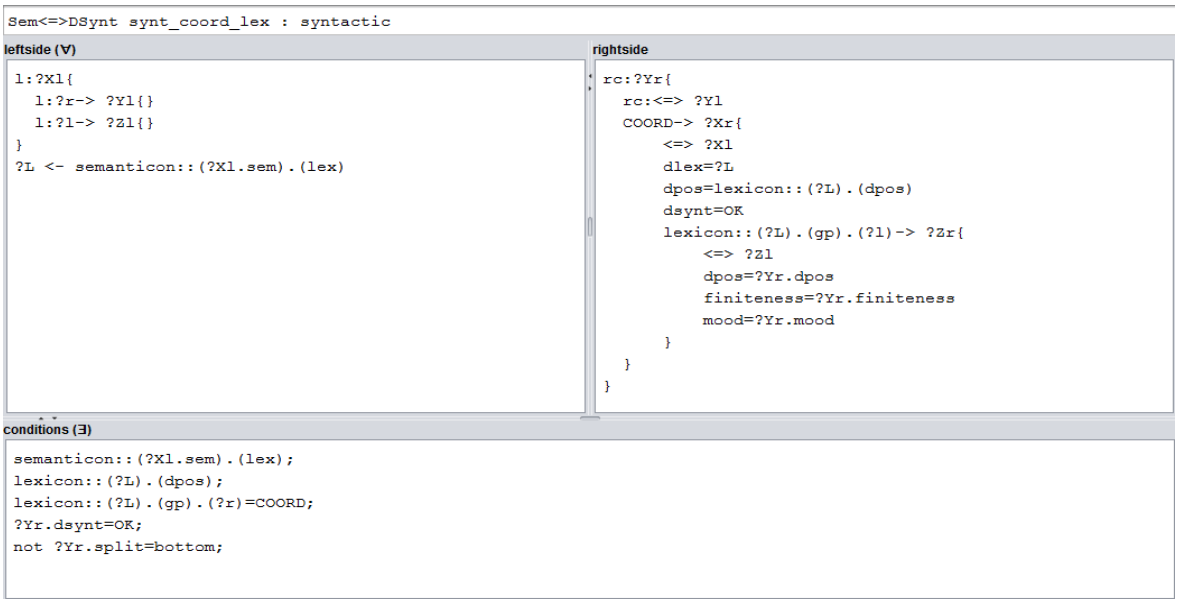
Conditions d'application

Avant d'appliquer les règles, le noeud *?Yr* doit déjà être lexicalisé *(?Yr.dsynt=OK*). Le lexème donné doit exister dans le semanticon *semanticon::(?Xl.sem).(lex).* Sa catégorie grammaticale doit être un verbe (*lexicon::(?L).(dpos)=V*). La relation *?r* entre n'est pas une coréférence.

2.1.3 Règle de coordination

Dans GÉCO, les mots-outils coordonnants *ir* ('et') et *bet* ('mais') font partie des structures sémantiques. Afin de traiter les phrases coordonnées, nous avons dû créer une nouvelle règle synt_coord_lex qui permette de réaliser des relations syntaxiques en RSyntP entre deux lexèmes ou syntagmes parallèles qui sont reliés par un sémantème coordonnant dans la RSem. La règle peut être illustrée comme suit :





Côté gauche

La règle instancie un nœud *?Xl* (mot-outil coordonant) qui possède deux arguments, les nœuds *?Yl* et *?Zl* (les éléments à coordonner). Ceux-ci sont reliés respectivement par les relations *?r* et *?l*. La règle instancie également une variable *?L* qui récupère la lexicalisation de sémantème coordonante (nœud *?Xl*) dans le lexicon (*ir* ou *bet*).

Côté droit

Le nœud *?Yr* existe déjà dans le graphe de sortie, il devient le correspondant du nœud *?Yl* (‘mange’). Ensuite, la règle crée une relation COORD entre *?Yr* et ?*Xr*, le nouveau correspondant du sémantème ‘et’ dans le graphe de sortie . Pour ce nœud, deux atributs sont crées : *dlex* (dont la valeur est *?L*=et) et *dpos* (la catégorie grammaticale qui est récupérée dans le lexicon, Adv). Le nœud est ensuite marqué comme lexicalisé (*dsynt=ok*). La commande *lexicon::(?L).(gp).(?l)* récupère la relation syntaxique de entre *?L* et *?Zl*, selon le régime de *?L*. Ensuite, le deuxième actant, le nœud *?Zr*, est créé. Il doit posséder certains des traits du premier actant de la coordination, *?Yr* : *dpos=?Yr.dpos* ; *finiteness=?Yr.finitenes ; mood=?Yr.mood*.

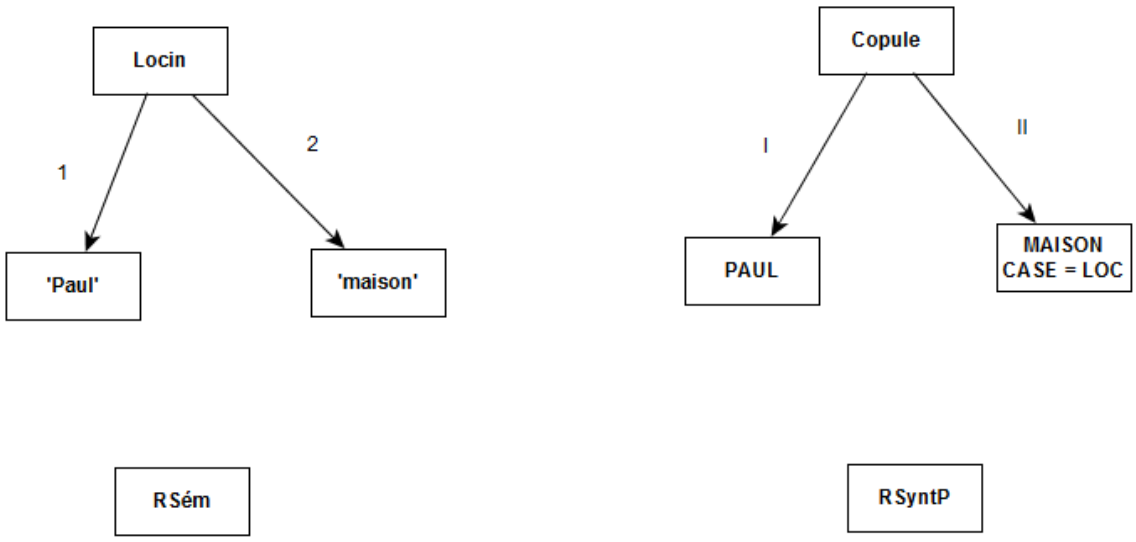
Conditions d’application

La condition *semanticon::(?Xl.sem).(lex)* spécifie que le sémantème doit exister dans le semanticon. Son *dpos* doit être défini dans le lexicon : *lexicon::(?L).(dpos).* Le premier actant du sémantème coordonant possède la relation COORD dans le gp du lexème : *lexicon::(?L).(gp).(?r)=COORD*. Le nœud *?Yr* doit être lexicalisé avant d’appliquer les règles : *?Yr.dsynt=OK*.

2.1.4 Cas sémantiques et FL Loc_{in}

Nous avons vu qu'en lituanien, il existe deux cas sémantiques : locatif et instrumental. Le cas locatif n'introduit pas de préposition tandis que le cas instrumental peut apparaître avec ou sans préposition.

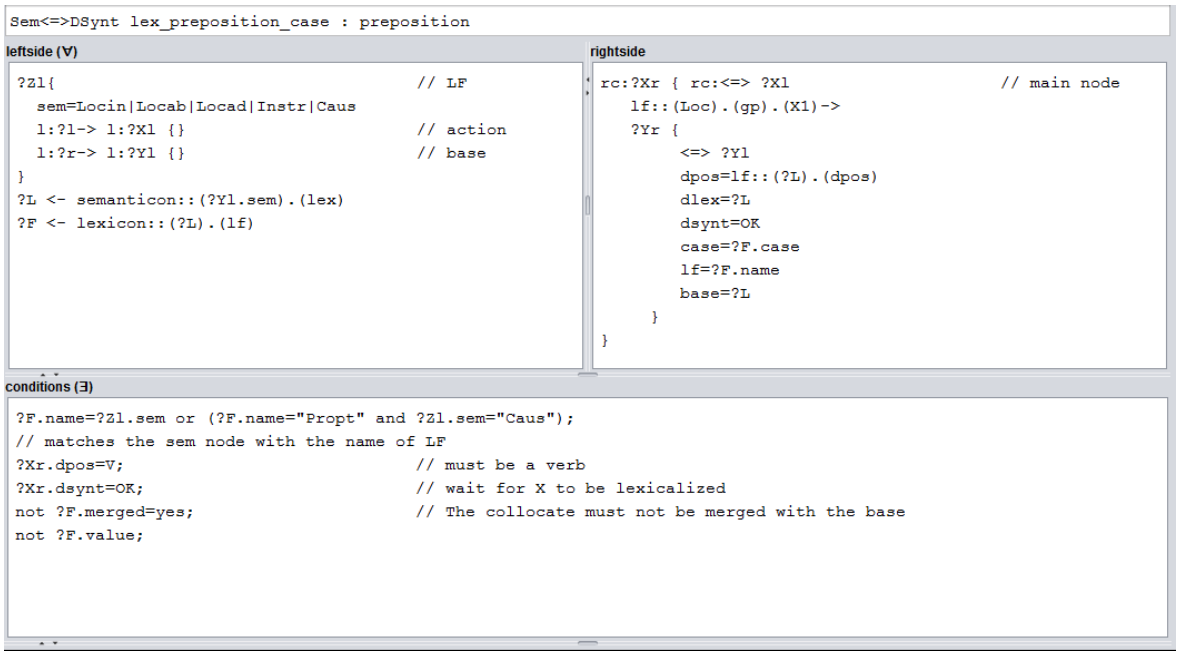
Dans les structures sémantiques, nous avons ajouté la FL Loc_{in}, qui exprime la localisation spatiale (parfois dans un sens métaphoriques). En lituanien, le Loc_{in} ne s'exprime pas par une préposition comme en français, mais par le cas sémantique locatif. Le fonctionnement de Loc_{in} en lituanien est donc différent de fonctionnement implémenté par Lambrey (2016). Le patron de Préposition, sous lequel est modélisé Loc_{in} ne pourra pas être appliqué pour générer nos phrases. De ce fait, nous avons dû appliquer quelques ajustements dans le *lexicon* et modifier les règles lex_preposition, et lex_preposition_case qui gèrent les FL de type prépositionnel. Le fonctionnement de la nouvelle règle de Loc_{in} est illustré ci-dessous :



Dans la classe NOUN du *lexicon*, nous avons ajouté les informations suivantes :

```
lf = {name = Locin
      case = LOC}
lf = {name = Locad
      case = ACC}
lf = {name = Locad
      value = į}
lf = {name = Instr
      value = su}
lf = {name = Instr
      case = INSTR}
```

Loc_{in} peut se réaliser uniquement par un cas. Les deux autres FL requièrent l'insertion d'une préposition en plus d'un cas. À l'aide de ces informations, les règles lex_preposition, et lex_preposition_case permettront respectivement d'insérer une préposition ou un cas spécifique selon la FL donnée.



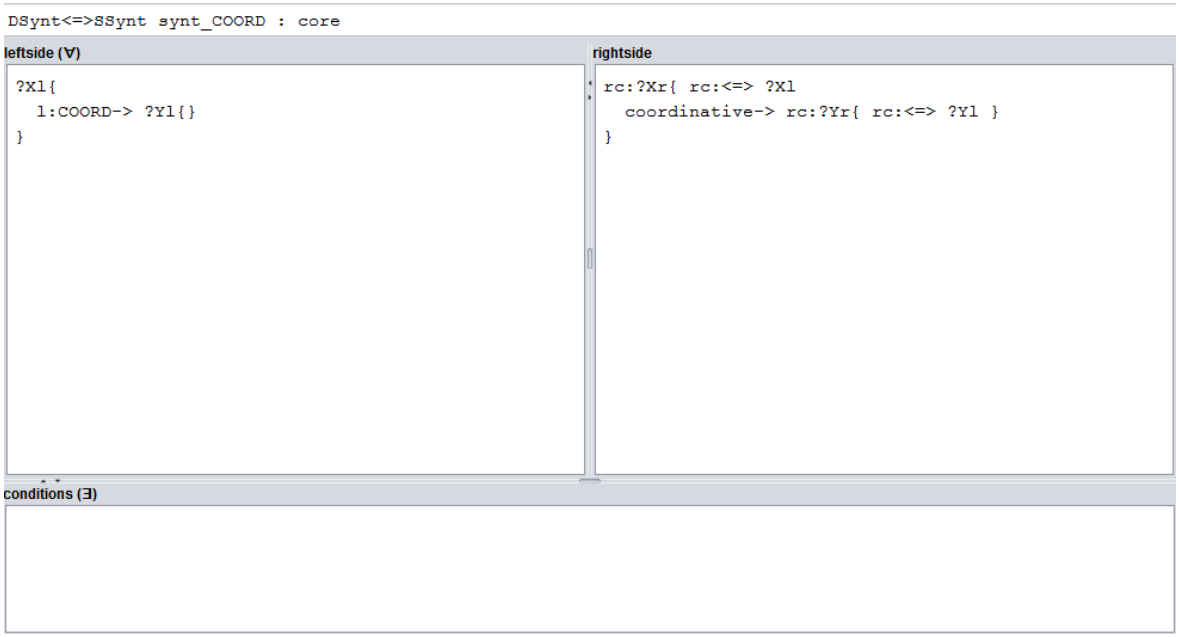
2.2 Module RSyntS

Nous avons apporté deux changements dans le module 25 : l'ajout de la règle de grammaire synt_COORD et un paquet lex_idiom, qui comporte deux règles de grammaire (lex_idiom_w1_w2 et lex_idiom_w1_w2_w3).

2.2.1 Règle de coordination

La règle synt_COORD est très simple. Elle récupère le produit de la règle de coordination en RSyntP et l'applique en RSyntS en réalisant la relation COORD par coordinative.

Voici la règle implémentée su MATE :



Côté gauche

La règle instancie un nœud ?Xl. Il possède un argument, ?Yl. La relation entre ces deux nœuds est spécifiée par la relation COORD en RSyntP.

Côté droit

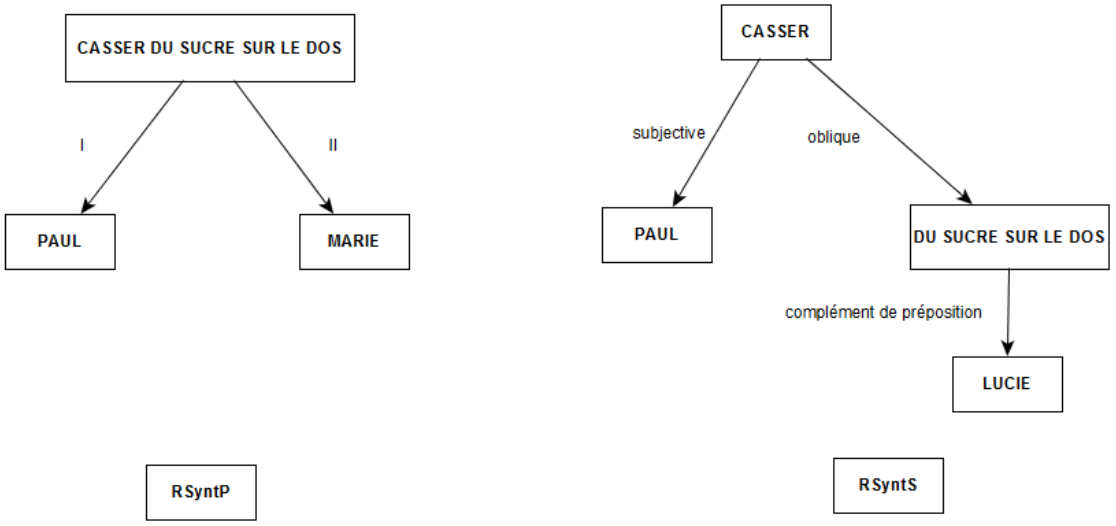
Dans le graphe de sortie, il existe un noeud *?Xr*, correspondant au nœud *?Xl* (*?Xr* <=> *?Xl*). Ensuite l'instruction *coordinative->* indique que la relation de type coordination est créée dans le graphe de sortie. Elle relie le nœud *?Xr* et *?Yr*, qui est le nouveau correspondant du nœud ?Yl spécifié dans le graphe de départ.

Conditions d'application

Cette règle peut être appliquée sans conditions d'application supplémentaires.

2.2.2 Traitement des locutions

Pour les locutions, nous avons créé des règles qui englobent tous les types de comportement de locutions. Voici le fonctionnement de la règle sous forme graphique :



Voici comment nous l'avons modélisée sur MATE :

```
DSynt<=>SSynt lex_idiom_w1_w2 : lex_idiom
leftside (∀)
l:?Xl
rightside
?W1 {
  <=> ?Xl
  dlex=?Xl.dlex
  slex=lexicon::(?Xl.dlex).idiom.w1
  dpos=lexicon::(?Xl.dlex).dpos
  spos=lexicon::(?Xl.dlex).spos
  ssynt=OK
  lexicon::(?Xl.dlex).idiom.r1-> ?w2 {
    slex=lexicon::(?Xl.dlex).idiom.w2
    dpos=lexicon::(lexicon::(?Xl.dlex).idiom.w2).dpos
    spos=lexicon::(lexicon::(?Xl.dlex).idiom.w2).spos
    ssynt=OK
  }
}
conditions (∃)
lexicon::(?Xl.dlex).idiom.type=w1_w2;
```

Côté gauche

La règle instancie un nœud *?Xl*. Le nœud fait référence à un nœud contenant une locution en RSyntP.

Côté droit

La règle crée un nouveau nœud *?Wl* dans le graphe de sortie. Il correspond au *?Xl*, comme défini par l'instruction *?W1 <=> ?Xl*. Le nœud réfère donc toujours à la locution entière. La règle crée également les attributs suivants :

- *dlex* dont la valeur est *?Xl.dlex* (la locution).
- *slex* dont la valeur est le segment principal de la locution (*w1*). Il est récupéré grâce à l'instruction *lexicon::(?Xl.dlex).idiom.w1*. La commande accède à l'entrée de la locution dans le lexicon et prend le mot *w1* (*casser*).
- Les catégories grammaticales en RSyntP (*dpos*) et RSyntS (*spos*) sont ensuite récupérées à l'aide des instructions *lexicon::(?Xl.dlex).dpos et spos=lexicon::(?Xl.dlex).spos*
- Le nœud est ensuite lexicalisé en SyntS (*ssynt=ok*). Le nœud est donc maintenant *casser*.

Ensuite, la règle procède à la création de la relation syntaxique de l'actant de *casser*. La relation est récupéré par la commande *lexicon::(?Xl.dlex).idiom.r1->* . La commande accède à l'entrée de la locution dans le lexicon et récupère la relation syntaxique *r1*, donc *oblique*.

Par la suite, la règle crée un nouveau nœud *?W2*. Celui-ci réfère au deuxième segment de la locution, donc *sur le dos*. Les attributs *slex*, *spos* et *dpos* sont créés grâce aux commandes permettant de récupérer les informations pertinentes dans le lexicon. Ensuite, le nœud est lexicalisé dans le graphe de sortie (*spos=ok*). Le nœud devient donc « *du sucre sur le dos* ».

Conditions d'application

Pour appliquer les règles, le nœud du graphe de départ doit référer à une locution de type w1_w2 dans le *lexicon*.

3. Temps et autres traits

Les informations enregistrées dans le *lexicon* permettent de gérer les actants syntaxiques et les cas des noms. Afin de compléter la description des noms dans les arbres syntaxiques générées, nous avons ajouté quelques traits pour chaque nom dans les représentations sémantiques des phrases. Les trois traits ajoutés sont :

- NUMBER : {SG;PL} (spécifie le nombre)
- GENDER : {FEM;MASC} (spécifie le genre)
- CLASS: proper_noun, (détermine le nom propre)

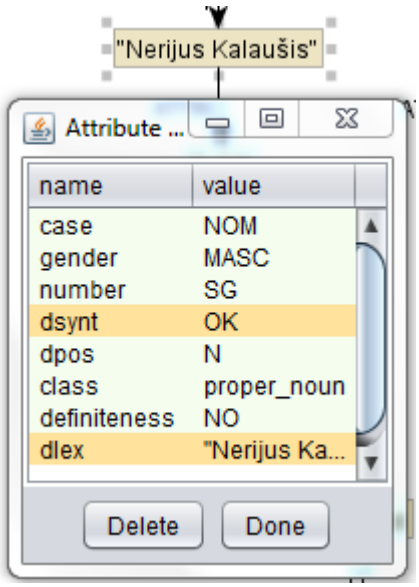
L'attribut CLASS déclanche l'application de la règle lex_special et bloque la règle lex_standard.

En plus des noms propres, l'attribut CLASS permet également de gérer les unités suivantes :

- les heures (hour)
- les dates (date)
- les unités (unit)
- les nombres (number)

Si nécessaire, elles peuvent être ajoutés dans les nœuds comportant les noms en sémantique.

Voici un exemple d'un nom propre généré en RSyntP. Le nœud comporte toutes les informations que nous lui avons attribuées.



En ce qui concerne les verbes, leur trait par défaut, si aucune autre valeur n'est spécifiée, est passé (*PAST*). Si dans la phrase il y a un verbe au temps futur, le trait *tense = FUT* est rajouté sur le nœud dans la représentation sémantique.

4. Travail futur

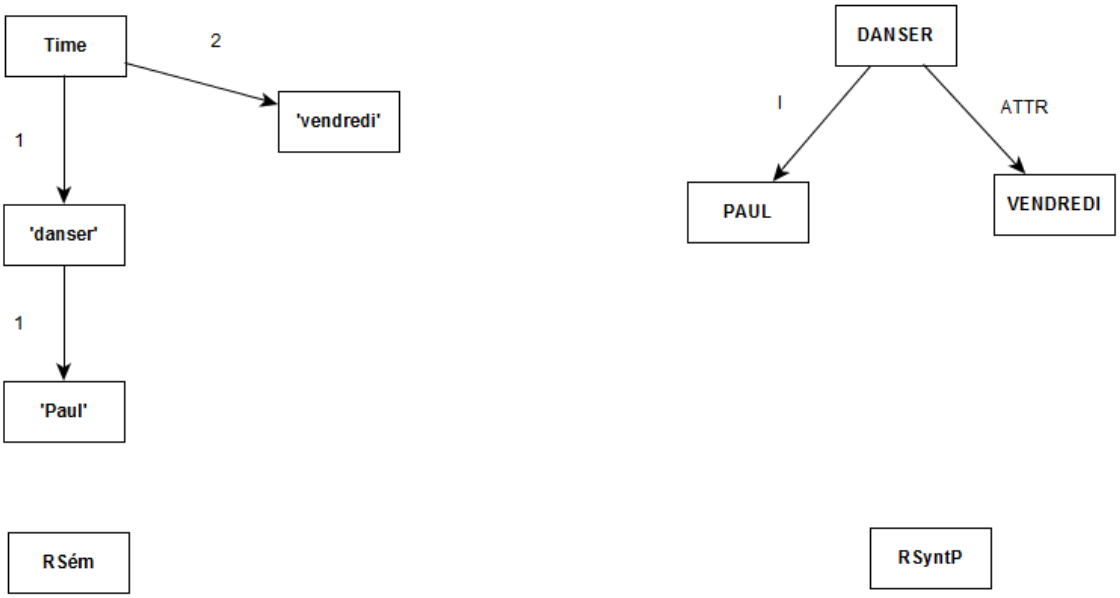
Dans cette section, nous présentons quelques phénomènes linguistiques que nous n'avons pas réussi à modéliser durant le stage par manque de temps. L'absence de certaines règles empêche l'arborisation correcte de certaines phrases. Les règles suivantes restent donc à dévélopper dans le futur : règle complète pour gérer le sémantème 'Time',

changement de cas de l'objet direct lors de la négation du verbe, et insertion de l'auxiliaire en RSyntS. Nous les décrivons brièvement dans les paragraphes qui suivent.

4.1 Règle Time

Comme nous l'avons expliqué dans le Chapitre 7, certaines structures sémantiques contiennent un sémantème générique 'Time'. Ce sémantème indique le temps d'un événement ou d'une action. Le premier actant est l'action ou l'évenement à situer tandis que le deuxième est le moment de l'action. Le deuxième actant peut être un nom, un adverbe ou un verbe.

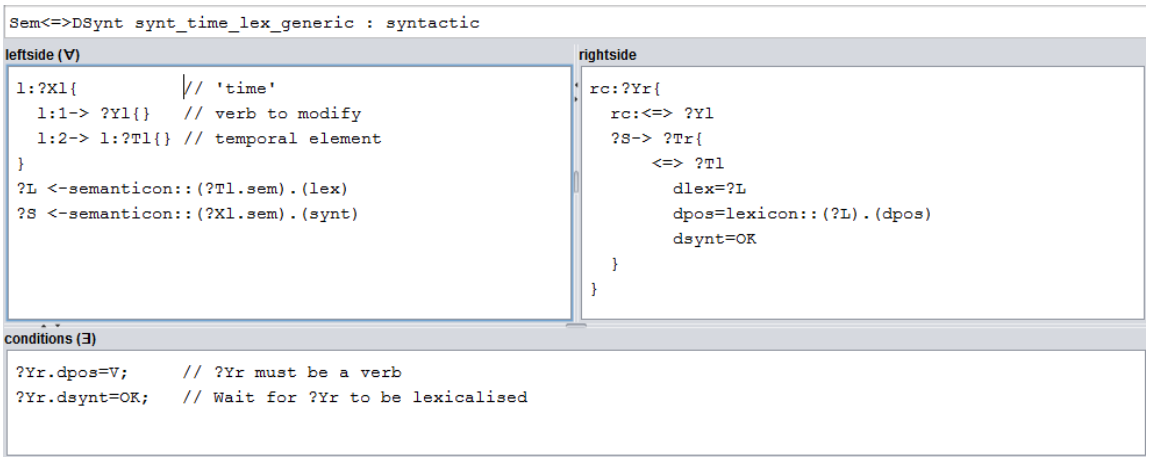
Afin de traiter les phrases où le deuxième actant de 'Time' est un nom ou un adverbe, nous avons créé une nouvelle règle de grammaire **synt_time_lex_generic** dans le paquet des règles d'arborisation du module RSyntP. Regardons maintenant le fonctionnement de cette règle.



Premièrement, nous avons ajouté le sémantème 'Time' dans le *semanticon*. Dans son entrée, nous avons indiqué que ce sémanème n'a pas de lexicalisation et sa réalisation syntaxique est de type modificateur : *synt = ATTR*.

```
Time {
  synt = ATTR
}
```

Ensuite, nous avons crée une nouvelle règle de grammaire dans MATE.



Côté gauche

La règle instancie un nœud *?Xl* (sémantème 'Time') qui possède deux arguments, les nœuds *?Yl* (verbe à modifier) et *?Tl* (élement temporel). Ceux-ci sont reliés respectivement par les relations *1* et *2*. La règle instancie une variable *?L* qui récupère la lexicalisation de l'élement temporel (nœud *?Tl*) dans le lexicon. La variable ?S est également instanciée. Elle permet de récupérer la valeur du *synt* dans le semanticon, donc *ATTR*.

Côté droit

Le noeud *?Yr* existe déjà dans le graphe de sortie et il est le correspondant de *?Yl* à gauche, ce qui est marqué par l'instruction *rc :?Yr ⇔?Yl*. Le noeud contenant le verbe est donc déjà créé dans le graphe de sortie. La règle crée un noeud *?Tr*, correspondant au *?Tl* du côté gauche (l'élement temporel). La relation entre les noeuds *?Yr* et *?Tr* est spécifiée par la relation *?S->*, dont la valeur, ATTR a été récupérée dans le côté gauche. Un attribut *dlex* est créé au noeud *?Tr*. Il correspond à la valeur de la variable *?L*, donc *vendredi*. Sa catégorie grammaticale en RSyntP (dpos) est récuperée par la commande *lexicon::(?L).(dpos).* Cette commande accède à l'entrée du lexème *?L* dans *lexicon* et reprend la valeur de son trait *dpos*. Ce noeud est ensuite marqué comme lexicalisé (*dsynt=ok*).

Conditions d'application

La condition *?Yr.dpos=V* indique que le premier actant du sémantème 'Time' doit être un verbe. Avant d'appliquer les règles, il doit être lexicalisé, *?Yr.dsynt=OK*.

Cette règle permet l'aborisation correcte de la Phrase 2, où l'élément temporel est un nom. Pourtant, elle est incomplète pour deux raisons. Premièrement, elle n'attribue pas

le cas au nom qui devrait être, en général, accusatif. Deuxièment, elle n'est pas capable de gérer les cas où le deuxième actant du sémantème est un verbe. Cela ne permet donc pas arboriser complètement les phrases 4 et 16.

4.2 Insertion de l'auxiliaire en RSyntS

Le lituanien possède un auxiliaire *būti* ('être'). L'auxiliaire devrait apparaître lors du passage de RSyntP à RSyntS. De ce fait, nous devrions créer une nouvelle règle qui permet de l'insérer dans les arbres syntaxiques générées en RSyntS.

4.3 Changement de cas lors de la négation

Dans le *lexicon*, nous avons indiqué que le cas de l'objet direct des verbes de type *verb_dt* est par défaut accusatif. Pourtant, nous n'avons pas prévu de gérer le changement de cas quand le verbe est précédé par une negation. Par exemple, le cas de l'objet direct de verbe *veikti* ('influencer') est accusatif. Pourtant, dans la phrase *prašymas neveikia sprendimo* ('la demande n'influence pas la décision'), le cas de 'décision' change pour le génitif. Nous ne pouvons pas généraliser ce comportement dans l'entrée de *veikti* ou autres entrées de *verbes_dt* dans le lexicon, car il s'applique spécifiquement lors de la négation. Il faudrait donc modéliser ce comportement de négation et créer une nouvelle règle générique dans le module RSyntP.

Synthèse

Dans le cadre de ce stage, nous avons implémenté un module du lituanien dans GÉCO, générateur automatique de texte multilingue. GÉCO est en cours de développement, et le lituanien fait partie des premières langues introduites dans le système. L'objectif de notre stage était de génerer les collocations et les arbres syntaxiques des phrases dans les niveaux RSyntP et RSyntS. Afin d'atteindre cet objectif, nous avons créé des ressources lexicographiques et développé des règles de grammaire génériques. Nous présentons ici une brève synthèse des tâches effectuées et des résultats obtenus.

Nous avons commencé par chercher un corpus de référence pour notre générateur. L'objectif principal de GÉCO étant la génération des collocations, nous avons choisi un texte contenant plusieurs collocations. Dans le cadre de ce stage, nous avons décidé de nous baser sur un texte suivi. L'entrée de notre système est donc constituée de représentations sémantiques de chacune des phrases du corpus, et la sortie attendue est l'ensemble des structures syntaxiques correspondantes.

Après avoir intégré les représentations sémantiques dans GÉCO, nous avons consacré la plus grande partie de notre travail au dévéloppment de ressources lexicographiques riches. Nous avons travaillé sur deux dictionnaires de GÉCO : *sémanticon* et *lexicon*. Dans le premier, nous avons listé les sémantèmes faisant partie des structures sémantiques. Pour chaque sémantème, nous avons listé ses lexicalisations possibles. Dans le deuxième dictionnaire, nous avons donné les propriétés lexicales et syntaxiques des lexèmes et nous avons fourni un répertoire des fonctions lexicales. Nous y avons également ajouté les traits grammaticaux spécifiques au lituanien : attribution des cas au noms selon le type de verbe.

Afin de pouvoir récupérer les propriétes lexicales et syntaxiques des lexèmes enregistrés dans le *lexicon*, nous avons repris quelques règles de grammaire de base de MARQUIS. Pourtant, elles n'étaient pas suffisantes pour couvrir les phénomènes linguistiques dont nous avions besoin pour générer les arbres syntaxiques cibles. De ce fait, nous avons eu l'occasion de développer différents types de règles : règles de lexicalisation, règles syntaxiques et règles de fonctions lexicales. Les règles que nous avons créées dans GÉCO sont génériques et peuvent s'appliquer à d'autres langues. Il est important de remarquer que nous n'avons créé qu'une seule règle spécifique au lituanien : une règle permettant de gérer les cas sémantiques, comme locatif et instrumental. Nous notons que cette règle est flexible est

pourra s'appliquer à d'autres langues à cas qui seront introduites plus tard dans le système. Tous les autres phénomènes linguistiques spécifiques à la langue sont décrits dans le *lexicon*.

La réalisation linguistique multilingue de GÉCO repose donc sur le principe de partage de la grammaire : l'implémentation de dictionnaires riches permet de concevoir des règles grammaticales plus génériques qui peuvent être utilisées pour plusieurs langues.

Afin de générer les collocations en lituanien, nous avons dû adapter les structures sémantiques des phrases pour y intégrer les composantes sémantiques de certaines FL. Grâce à l'implémentation quasi-exhaustive des FL dans GÉCO, nous avons pu générer des collocations de plusieurs types. L'intégration des FL nous a également permis de générer plusieurs paraphrases, augmentant la diversité et le nombre des arbres syntaxiques générés.

Au total, nous avons généré 132 arbres syntaxiques pour 16 phrases de départ, soit une moyenne de 7 paraphrases par structure d'entrée.

Bibliographie

Alonso Ramos, M., & Tutin, A. (1996). A Classification and Description of Lexical Functions for the Analysis of their Combinations. In L. Wanner (Éd.), *Lexical Functions in Lexicography and Natural Language Processing* (p. 147-168). Amsterdam/Philadelphia: John Benjamins Publishing.

Apresjan, J. D., Boguslavsky, I., & Tsinman, L. L. (2007). Lexical Functions in Actual NLP-Applications. In L. Wanner (Éd.), *Selected Lexical and Grammatical Issues in Meaning-Text Theory. In honour of Igor Mel'cuk* (Vol. 84, p. 199-230). John Benjamins.

Avgustinova, T., & Uszkoreit, H. (2000). An ontology of systematic relations for a shared grammar of Slavic. In *Proceedings of the International Conference on Computational Linguistics* (Vol. 1, p. 28-34). Saarbrücken, Germany.

Baez B., Portet F., Caffiau S. & Garbay C. (2015) Generating récit from sensor data: evaluation of coherence and preliminary experiments with GPS data, In 5th *International Workshop on Computational Models of Narrative*, pages 1–10.

Barzilay, R., and McKeown, K. (2001). Extracting paraphrases from a parallel corpus. In 39th Annual Meeting of the Association for Computational Linguistics, 50–57.

Barzilay R., Lee L. (2003). Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL-HLT*, 16–23.

Bateman, J. (1997). Sentence generation and systemic grammar: an introduction. *Iwanami Lecture Series : language Sciences, 8*.

Bateman, J. A. (1997). Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, *3*(1), 15–55.

Bateman, J. A., Kruijff-Korbayová, I., & Kruijff, G.-J. (2005). Multilingual Resource Sharing Across Both Related and Unrelated Languages: An Implemented, Open-Source Framework for Practical Natural Language Generation. *Res Lang Comput Research on Language and Computation*, *3*(2-3), 191-219.

Bateman, J., Matthiessen, C., Nanri, K., & Zeng, L. (1991). The Re-use of Linguistic Resources Across Languages in Multilingual Generation Components. In *Proceedings*

of the 12th International Joint Conference on Artificial Intelligence - Volume 2 (p. 966–971). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Benson, M. (1990). Collocations and General-purpose Dictionaries. *Int J Lexicography International Journal of Lexicography*, *3*(1), 23-34.

Boguslavsky, I., Iomdin, L., & Sizov, V. (2004). Multilinguality in ETAP-3: Reuse of Lexical Resources. In *Proceedings of the Workshop on Multilingual Linguistic Ressources* (p. 7–14). Stroudsburg, PA, USA: Association for Computational Linguistics.

Bohnet, B. (2006). Development Environment forGraph/Tree Transducer Grammar and other linguistic Ressources.

Bohnet, B., Langjahr, A., & Wanner, L. (2000). A Development Environment for MTT-Based Sentence Generators. In *Proceedings of the XVI SEPLN Conference*. Vigo, Spain.

Bohnet, B., Lareau, F., Wanner, L., & others. (2007). Automatic production of multilingual environmental information. In *Proceedings of the 21st Conference on Informatics for Environmental Protection (EnviroInfo-07), Warsaw, Poland*.

Bohnet, B., & Wanner, L. (2010). Open source graph transducer interpreter and grammar development envrionment. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*. Valletta, Malta: European Language Resources Association (ELRA).

Bollmann, M. (2011). Adapting SimpleNLG to German. In *Proceedings of the 13th European Workshop on Natural Language Generation* (p. 133–138). Stroudsburg, PA, USA: Association for Computational Linguistics.

Bonami, O., & Boyé, G. (2010). La morphologie flexionnelle est-elle une fonction ? In I. Choi-Jonin, M. Duval, & O. Soutet (Éd.), *Typologie et comparatisme. Hommages offerts à Alain Lemaréchal.* (p. 21-35). Louvain, Belgique: Peeters.

Bresnan, J. (2001). *Lexical-Functional Syntax*. Oxford: Blackwell.

Butt, M. (1999). *A grammar writer's cookbook*. Stanford, Calif.: CSLI Publications.

Butt, M., Dyvik, H., King, T. H., Masuichi, H., & Rohrer, C. (2002). The Parallel Grammar Project. In *Proceedings of the 2002 Workshop on Grammar Engineering and*

Evaluation (Vol. 15, p. 1–7). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/1118783.1118786

Dale, R., Moisl, H. L., & Somers, H. L. (2000). *Handbook of natural language processing*. New York: Marcel Dekker.

Dalrymple, M. (2006). Lexical-Functional Grammar. In *Encyclopedia of Cognitive Science*. John Wiley & Sons, Ltd.

Danlos, L. (1987a). The linguistic basis of text generation. In *Proceedings of the third conference on European chapter of the Association for Computational Linguistics* (p. 1-1). Association for Computational Linguistics. https://doi.org/10.3115/976858.976859

Danlos, L. (1987b). *The Linguistic Basis of Text Generation*. Cambridge University Press.

Danlos, L. (1998). G-TAG : un formalisme lexicalisé pour la génération de textes inspiré de TAG. *Traitement Automatique des Langues, 39*(2), 28 p.

Danlos, L., & Roussarie, L. (2000). La génération automatique de textes. In J.-M. Pierrel (Éd.), *Ingénierie des langues*. Paris: Hermès.

Davey, A. (1979). Discourse Production. Edinburgh University Press, Edinburgh, Scotland.

Fontenelle T. (1997a) Dictionnaires électroniques et relations lexicales : une comparaison entre quelques programmes européens. *Revue Française de Linguistique Appliquée*, II-1:65–77.

Fontenelle T. (19972). Turning a Bilingual Dictionary into a Lexical Semantic Database. Max Niemeyer, Tübingen.

Goldberg E., Driedger N. & Kittredge R. (1994). Using natural language processing to produce weather forecasts, IEEE Expert , 9 (2) :45 -53.

Goldman N. (1975). Conceptual generation. In Roger C. Shank and Christopher K. Riesbeck, editors, Conceptual Information Processing, American Elsevier, New York

Grimes J., (1990) Inverse Lexical Functions. Dans James Steele, dir., *Meaning Text Theory: Linguistics, Lexicography and Implications*, pages 350–364. Ottawa University Press, Ottawa.

Haller S. & Di Eegenio B. (2003). Minimal text structuring to improve the generation of feedback in intelligent tutoring systems. In FLAIRS 2003, the *16th International Florida AI Research Symposium*, St. Augustine, FL.

Heid, U., & Raab, S. (1989). Collocations in Multilingual Generation. In *Proceedings of the Fourth Conference on European Chapter of the Association for Computational*

Linguistics (p. 130–136). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/976815.976833

Hellwig, H. J. Heringer, & H. Lobin (Éd.), *Dependency and Valency. An International Handbook of Contemporary Research* (Vol. 1, p. 546-570). Berlin - New York: W. de Gruyter.

Heylen, D., Maxwell, K. G., & Verhagen, M. (1994). Lexical Functions and Machine Translation. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 2* (p. 1240–156 1244). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/991250.991354

Iordanskaja, L., Kittredge, R., and Polguère, A. 1991. Natural Language Generation *in Artificial Intelligence and Computational Linguistics*. Kluwer Academic. Chapter Lexical selection and paraphrase in a meaning-text generation model.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B., & Polguère, A. (1992). Generation of Extended Bilingual Statistical Reports. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 3* (p. 1019–1023). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/993079.993120

Iordanskaja, L., Kim, M., & Polguère, A. (1996). Some Procedural Problems in the Implementation of Lexical Functions for Text Generation. In L. Wanner (Éd.), *Lexical Functions in Lexicography and Natural Language Processing* (p. 279-298). Amsterdam/Philadelphia: John Benjamins Publishing.

Jousse, A.-L. (2003). *Normalisation des fonctions lexicales*. Paris 7.

Jousse, A.-L. (2010). *Modèle de structuration des relations lexicales fondé sur le formalisme des fonctions lexicales*. Paris 7.

Kahane, S. (2003). The Meaning Text Theory. In V. Agel, L. Eichinger, H.-W. Eroms, P.

Kahane, S., (2001a) Grammaires de dépendance formelles et théorie Sens-Texte, 2001, Actes TALN.

Kahane, S., & Polguère, A. (2001). Formal foundation of lexical functions. In *Proceedings of ACL/EACL 2001 Workshop on Collocation* (pp. 8–15).

Kim, R., Dalrymple, M., Kaplan, R., Holloway King, T., Masuichi, H., & Ohkuma, T. (2003). Multilingual Grammar Development via Grammar Porting. In *Proceedings of the ESSLLI Workshop on Ideas and Strategies for Multilingual Grammar Development*. Vienna, Austria. 157

Kittredge, R., Iordanskaja, L., & Polguère, A. (1988). Multi-Lingual Text Generationand the Meaning-Text Theory. In *Conference on Theoretical and Methodological Issues in Machine Translation of Natur al Languages*. Pittsburg, PA, USA.

Lambrey, F. (2016). *Implémentation des collocations pour la réalisation de texte multilingue*. Université de Montréal.

Lambrey, F., & Lareau, F. (2015). Le traitement des collocations en génération de texte multilingue. In *Actes de TALN 2015* (p. 579-585). Caen.

Lareau, F., Dras, M., Börschinger, B., & Dale, R. (2011). Collocations in Multilingual Natural Language Generation: Lexical Functions meet Lexical Functional Grammar. In *Proceedings of ALTA'11* (p. 95-104). Canberra.

Lareau, F., Dras, M., Börschinger, B., & Turpin, M. (2012). Implementing lexical functions in XLE. In *Proceedings of LFG12* (p. 362-382). Denpasar, Indonesia.

Lareau, F., & Wanner, L. (2007). Towards a Generic Multilingual Dependency Grammar for Text Generation. In *Proceedings of the GEAF07 Workshop* (p. 203-223). Stanford: CSLI.

Marcu D., Carlson D., Watanabe M. (2000). An Empirical Study in Multilingual Natural Language Generation: What Should A Text Planner Do*? In Proceedings of the 1st International Conference on Natural language Generation (INLG'00*), pages 17–23.

Martinez, Eds., Proceedings of the Australasian Language Technology Association Workshop, p. 95–104, Canberra.

Matthiessen, C. M. I. M., & Bateman, J. A. (1991). *Text generation and systemic-functional linguistics: experiences from English and Japanese*. London: Pinter.

Maxwell, K., & Heylen, D. (1994). Lexical Functions and the Translation of Collocations. In *Euralex* (p. 298-305).

Mel'čuk, I. (1967). Ordre des mots en synthèse automatique des textes russes. T.A. Informations, 8:2, 65-84.

Mel′čuk, I. (1973). Linguistic Theory and « Meaning-Text » Type Models. In R. Bogdan & I. Niiniluoto (Éd.), *Logic, language and Probability* (p. 223-235). Dordrecht-Boston: Reidel.

Mel′čuk, I. A. (1988). *Dependency syntax: theory and practice*. Albany: State University Press of New York.

Mel′čuk I., (1993). Cours de morphologie générale, vol. 1. Presses universitaires de Montréal.

Mel′čuk, I. (1996). Lexical Functions: A Tool for the Description of Lexical Relations in a Lexicon. In L. Wanner (Éd.), *Lexical Functions in Lexicography and Natural Language Processing* (p. 37-102). Benjamins.

Mel′čuk, I. (1997) Vers une linguistique Sens-Texte. Leçon inaugurale. Collège de France, Paris, 10 janvier 1997.

Mel′čuk, I. (1998). Collocations and lexical functions. *en Anthony Paul COWIE (ed.)(2001 [1998])*, 23–54.

Mel′čuk, I. (2003). Les collocations : définition, rôle et utilité. In F. Grossmann & A. Tutin (Éd.), *Les collocations : analyse et traitement*. Amsterdam: De Werelt.

Mel′čuk, I. (2004). Les verbes supports sans peine. *Lingvisticae Investigationes, 27*(2), 203-217.

Mel′čuk, I. (2013). Tout ce que nous voulions savoir sur les phrasèmes, mais ... *Cahiers de lexicologie, 102,* 129-150.

Mel′čuk, I., Arbatchewsky-Jumarie, N., & Clas, A. P. (1999). *DEC: dictionnaire explicatif et combinatoire du français contemporain : recherches lexico-sématiques IV*. Montréal: Presses de l'Université de Montréal.

Mel′čuk, I., Clas, A., & Polguère, A. (1995). *Introduction à la lexicologie explicative et combinatoire*. Bruxelles: Duculot.

Meyer I., et Steele J. (1990) The Presentation of an Entry and Super-entry in an Explanatory Combinatorial Dictionary of English. Dans James Steele, dir., *Meaning-Text Theory, Linguistics, Lexicography and Implications*, pages 62–94. University of Ottawa Press, Ottawa.

Milićević, J. (2007). La paraphrase: modélisation de la paraphrase langagière, Peter Lang. Bern.

Polguère A., (1998a) La Théorie Sens-Texte", 1998a, Dialangue, Vol. 8-9, Université du Québec à Chicoutimi.

Polguère, A. (1998b). Pour un modèle stratifié de la lexicalisation en génération de texte. *Traitement Automatique des Langues (TAL), 39*(2), 57–76. 160

Polguère, A. (2000). A « natural » lexicalization model for language generation. In *Proceedings of the Fourth Symposium on Natural Language Processing 2000* (Vol. 10-12, p. 37-50). Chiangmai, Thailand,.

Polguère, A. (2007). Lexical Function Standardness. In L. Wanner (Éd.), *selected lexical and grammatical issues in the Meaning-Text Theory: In honour of Igor Mel'cuk* (p. 43-96). John Benjamins Publishing.

Polguère, A. (2012). Propriétés sémantiques et combinatoires des quasi-prédicats sémantiques. Scolia, Université des sciences humaines Strasbourg, 2012, 26, pp.131-152.

Portet F., Reiter E., Gatt A., Hunter, J., Sripada, S., Freer, Y., & Sykes, C. (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence,* 173(7-8):789–816.

Rassinoux A., Baud R., Rodriques J.M., Lovis C., 2007. Coupling Ontology Driven Semantic Representation with Multilingual Natural Language Generation for Tuning International Terminologies. In *Proceedings of the 12th World Congress on Health (Medical) Informatics (MEDINFO'07)*, pages 555–559, Brisbane.

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3*(1), 57-87.

Reiter, E., & Dale, R. (2000). *Building Natural Language Generation Systems*. New York, NY, USA: Cambridge University Press.

Roussarie, L. (2000). *Un modele theorique d'inference de structures semantiques et discursives dans le cadre de la generation automatique de textes*. Paris 7.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., & Flickinger, D. (2002). Multiword Expressions: A Pain in the Neck for NLP. In A. Gelbukh (Éd.), *Computational Linguistics and Intelligent Text Processing* (p. 1-15). Mexico city: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45715-1_1

Santaholma, M. (2008). Multilingual Grammar Resources in Multilingual Application Development. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks* (p. 25–32). Stroudsburg, PA, USA: Association for Computational Linguistics.

Sowa, J. F. (2000). *Knowledge representation: logical, philosophical, and computational foundations*. Pacific Grove: Brooks/Cole.

Stede, M. (1996). Lexical options in multilingual generation from a knowledge base. In G. Adorni & M. Zock (Éd.), *Trends in Natural Language Generation An Artificial Intelligence* 161 *Perspective* (p. 222-237). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-60800-1_32

Steele J. (1986). A Lexical Entry for an Explanatory Combinatorial Dictionary of English (hope II.1). Dictionaries, 8:1–54.

Steinlin, J. (2003). *Générer des collocations*. Paris 7, Paris.

Wanner, L. (Éd.). (1996). *Lexical Functions in Lexicography and Natural Language Processing*. Amsterdam/Philadelphia: John Benjamins Publishing.

Wanner, L. (Éd.). (2007). *Selected Lexical and Grammatical Issues in the Meaning–Text Theory. In honour of Igor Mel'cuk*. John Benjamins.

Wanner, L., Bohnet, B., Bouayad-Agha, N., Lareau, F., Lohmeyer, A., & Nicklaß, D. (2007). On the Challenge of Creating and Communicating Air Quality Information. In *Proceedings of ISESS 2007* (p. 356-367).

Wanner, L., Bohnet, B., Bouayad-Agha, N., Lareau, F., & Nicklaß, D. (2010). Marquis: Generation of User-Tailored Multilingual Air Quality Bulletins. *Applied Artificial Intelligence*, *24*(10), 914-952. https://doi.org/10.1080/08839514.2010.529258

Wanner, L., & Lareau, F. (2009). Applying the Meaning-Text Theory Model to Text Synthesis with Low- and Middle-Density Languages in Mind. In S. Nirenburg (Éd.), *Language Engineering for Lesser-Studied Languages*. IOS Press.

Wanner, L., Nicklaß, D., Bouayad-Agha, N., Bohnet, B., Bronder, J., Ferreira, F., others. (2007). From Measurement Data to Environmental Information: MARQUIS-A Multimodal AiR QUality Information Service for the General Public. In *Poceedings of ISESS 2007*. Prague.

Žolkovsky A. et Mel′čuk, I. (1966) O sisteme semanticeskogo sinteza. I. stroenie slovarja. Naucno-texniceskaya informacija, 11:48–55.

Žolkovsky A. et Mel′čuk, I. (1967) semanticeskom sinteze. Problemy kibernetiki, 19:177–238.

Žolkovsky A. et Mel′čuk, I. (1967). Sur la synthèse sémantique. T. A. Informations, 2:1–85

Table des illustrations

MOTS-CLÉS : génération automatique de texte multilingue, lexicalisation, collocations, lituanien

RÉSUMÉ

Le premier objectif de ce mémoire est d'intégrer un module de génération du lituanien dans GÉCO, un générateur automatique de texte multilingue exploitant des ressources linguistiques riches fondées sur le principe du partage de grammaire. Ce générateur est en cours de développement à l'Observatoire de linguistique Sens-Texte de l'Université de Montréal. Il est basé sur la grammaire de MARQUIS et vise à implémenter de manière exhaustive les collocations dans le cadre de la génération automatique de texte multilingue. Notre deuxième objectif est de générer des collocations en lituanien en testant ainsi la performance du système de GÉCO.

KEYWORDS : multilingual natural language generation, lexicalisation, collocations, lithuanian

ABSTRACT

The primary goal of this thesis is to integrate a module of text generation in Lithuanian language in GÉCO, multilingual natural language generator based on rich lexicographic resources and principles of grammar sharing. This generator is under development at the Observatoroire de linguistique Sens-Texte of the University of Montreal. It is based on the grammar of MARQUIS and aims to implement a large set of collocations within the framework of the multilingual natural language generation. Our second objective is to generate collocations in Lithuanian by thus testing the performance of GÉCO.