Vers un système générique de réécriture de graphes pour l'enrichissement de structures syntaxiques.

Corentin Ribeyre[1, 2]
(1) Université Paris 7 Diderot, 75013 PARIS
(2) INRIA Paris-Rocquencourt, Rocquencourt BP 105 78153 LE CHESNAY
corentin.ribeyre@inria.fr

RÉSUMÉ

Ce travail présente une nouvelle approche pour injecter des dépendances profondes (sujet des verbes à contrôle, partage du sujet en cas d'ellipses, ...) dans un corpus arboré présentant un schéma d'annotation surfacique et projectif. Nous nous appuyons sur un système de réécriture de graphes utilisant des techniques de programmation par contraintes pour produire des règles génériques qui s'appliquent aux phrases du corpus. Par ailleurs, nous testons la généricité des règles en utilisant des sorties de trois analyseurs syntaxiques différents, afin d'évaluer la dégradation exacte de l'application des règles sur des analyses syntaxiques prédites.

ABSTRACT

Towards a generic graph rewriting system to enrich syntactic structures

This work aims to present a new approach for injecting deep dependencies (subject of control verbs, subject sharing in case of ellipsis, ...) into a surfacic and projective treebank. We use a graph rewriting system with constraint programming techniques for producing generic rules which can be easily applied to a treebank. Moreover, we are testing the genericity of our rules by using output of three different parsers to evaluate how the rules behave on predicted parse trees.

MOTS-CLÉS : réécriture de graphes, évaluation de shéma d'annotations, parsing, analyse en syntaxe profonde.

KEYWORDS: graph rewriting system, annotation schemes evaluation, deep syntax parsing.

Introduction

Le French Treebank (FTB) est un corpus arboré du français, qui, comme bien d'autres, se veut neutre d'un point de vue théorique. De fait son schéma d'annotation est à mi-chemin entre les constituants et les dépendances. Le treebank est non configurationnel [1]. C'est pourquoi le schéma d'annotation choisi est assez plat et essentiellement surfacique (Abeillé *et al.*, 2003). On cherche alors à obtenir des représentations plus profondes, telles que le sujet des infinitives, l'antécédent des relatifs, ou encore les sujets elliptiques et le véritable sujet des verbes à contrôle.

Pour ce faire, nous utilisons la réécriture de graphes (Löwe *et al.*, 1993; Geiss *et al.*, 2006) – domaine selon lequel on modifie des graphes au moyen de règles. On recherche une sous-

1. Les structures arborescentes ne sont pas suffisantes pour distinguer un syntagme nominal objet d'un ajout, par exemple.

structure dans le graphe que l'on veut transformer et on applique des modifications à cette structure : ajout d'arcs, de nœuds, modifications des structures de traits sur les nœuds et les arcs, etc. C'est une discipline émergente en traitement automatique des langues, que l'on utilise, par exemple, pour transformer des dépendances surfaciques en graphes sémantiques (Bonfante *et al.*, 2011a). Enfin, la réécriture de graphes peut être mise au profit du passage d'un schéma d'annotation à un autre.

Cependant, lorsque l'on souhaite transformer des structures linguistiques complexes, on se retrouve confronté à deux problèmes : d'une part, la diversité des configurations syntaxiques entraîne une augmentation importante du nombre de règles et, d'autre part, la présence de phénomènes syntaxiques dits « non locaux » demandent des règles complexes, voire récursives, dont l'application est régie par un ordre strict, comme c'est le cas dans le système de réécriture de graphes *Grew* de Bonfante *et al.* (2011b). Par exemple, pour retrouver les antécédents des pronoms relatifs dans le FTB, il faut suivre une chaîne de dépendances plus ou moins longue, afin de remonter du pronom relatif vers l'antécédent. On a aussi les cas de coordinations et notamment de coordinations elliptiques, tels que l'ellipse du sujet où il y a partage du sujet entre les deux conjoints coordonnés. Ajouté à cela le fait qu'il puisse y avoir des constructions elliptiques avec plusieurs modaux entraînant un contrôle, et le partage du sujet devient plus complexe ; c'est notamment le cas dans *Jean pense partir aujourd'hui et rentrer demain*, où la configuration initiale donne *Jean*, sujet de *pense*, or il est aussi le sujet de *partir* et de fait, celui de *rentrer*. En somme, l'écriture de règles pour transformer des graphes peut se révéler longue et difficile, et leur nombre croître rapidement, rendant un tel système difficile à maintenir.

L'approche que nous présentons ici est issue de travaux antérieurs (Ribeyre, 2012; Ribeyre *et al.*, 2012). Nous avons mis en place un système de réécriture qui procède en deux temps : la première technique est fondée sur une approche qui, étant donné un motif de graphe et un graphe de remplacement, tente de retrouver le motif dans un graphe donné et de le remplacer par le graphe de remplacement. Cette approche est largement documentée dans la littérature sur les graphes (Löwe *et al.*, 1993; Geiss *et al.*, 2006). Mais nous apportons une seconde approche, fondée sur la programmation par contrainte (Fruewirth et Abdennadher, 2003), et dite de « propagation de contraintes » sur les arcs. Nous attachons des contraintes sur les arcs du graphe à réécrire et en fonction de celles-ci, des modifications sont effectuées.

Pour valider notre approche, nous avons appliqué une série de règles à deux phénomènes de la syntaxe : (i) le contrôle obligatoire ; (ii) la coordination à ellipse du sujet. Nous cherchons à ajouter les informations manquantes sur un treebank et à tester ensuite leur généricité sur les arbres syntaxiques produits par un analyseur syntaxique (parser) symbolique et deux parseurs statistiques entraînés sur le French Treebank. En effet, les parseurs ne produisant pas toujours d'analyses exactes, nous voulions pouvoir mesurer l'impact des mêmes règles sur des structures syntaxiques plus bruitées. L'idée sous-jacente permet d'ouvrir la voie à la correction d'analyses syntaxiques avec des règles simples.

Nous faisons tout d'abord un rapide état de l'art en matière de réécriture de graphes appliquée à l'enrichissement de corpus, puis présentons notre système de réécriture et plus particulièrement le système de propagation de contraintes, pour ensuite appliquer ce système à la réécriture de phrases issues du FTB (Abeillé *et al.*, 2003) et du SequoiaBank (Candito et Seddah, 2012), phrases qui présentent des phénomènes de contrôle et de coordination à ellipses sujet. Enfin, nous appliquons nos règles aux analyses syntaxiques produites par un parseur symbolique, FrMG (Villemonte de La Clergerie, 2005), et deux parseurs statistiques : le Malt parseur (Nivre

et al., 2006) et le MST parser (McDonald *et al.*, 2005b,a), afin de tester leur généricité. Enfin, nous conclurons sur les perspectives et les autres applications possibles de ce genre de système.

1 Etat de l'art

La réécriture de graphes appliquée à la syntaxe profonde et à l'interface syntaxe-sémantique est un domaine assez jeune. A notre connaissance, seul le système de réécriture *Grew* (Marchand *et al.*, 2010; Bonfante *et al.*, 2010, 2011b) a été utilisé dans ce but.

Grew est un système de réécriture qui s'appuie sur une hiérarchisation en modules pour traiter des phénomènes syntaxiques. Chaque module représente un phénomène bien particulier. De plus, les modules assurent la terminaison du système et une forme de confluence. Cependant, la hiérarchisation des phénomènes syntaxiques en modules est complexe, car les phénomènes interagissent et il devient souvent compliqué de gérer ces interactions, notamment avec les cas de coordinations elliptiques du sujet et de contrôle, mais aussi lors de la recherche de l'antécédent du pronom relatif, comme nous le précisions en introduction. De fait, certaines règles peuvent être présentes dans plusieurs modules, rendant la gestion d'un phénomène parfois éclaté au sein des différents modules.

L'article Bonfante *et al.* (2011b) traite un grand nombre de phénomènes syntaxiques :
– Sujet des participiales, des infinitives et des adjectifs
– Tough movement
– Verbes à contrôle
– Coordinations elliptiques du sujet
– Antécédent des pronoms relatifs
Le système est particulièrement couvrant et le corpus arboré ainsi enrichi se rapproche de la syntaxe profonde. Cependant l'article ne présente aucune évaluation quant à la précision du système et à sa capacité à rendre des analyses correctes.

Nous avons opté pour une évaluation sur des cas complexes et présentant souvent des interactions intéressantes sans pour autant essayer de couvrir un nombre aussi important de phénomènes. Par ailleurs, il nous a semblé tout aussi important de voir à quel point les règles mises au point sur un corpus donné pouvaient encore s'appliquer sur des analyses prédites par des parseurs divers.

2 Présentation du système de réécriture de graphes

OGRE, pour *Optimized Graph Rewriting Engine*, est un système de réécriture de graphes orienté TAL qui permet d'assurer un nombre réduit de règles faciles à maintenir sur des exemples qui peuvent être complexes en terme de réécriture de structures linguistiques. Une description formelle du système peut être trouvée dans (Ribeyre, 2012; Ribeyre *et al.*, 2012), nous rappelons ici les différents types de contraintes et les illustrons sur des exemples linguistiques afin de montrer leurs applications et leurs avantages.

On définit $e = (x \xrightarrow{l} y, C, H)$ un arc étendu qui peut porter un ensemble C potentiellement vide de contraintes et une liste H potentiellement vide représentant un historique formé par des pairs

de nœuds (x', y'). Cet historique nous permet de retracer les changements effectués entre x' et y'. D'autre part, on définit aussi $\mathscr{L}$ qui est l'ensemble des étiquettes possibles sur un arc. On considère alors trois types de contraintes :

– Une contrainte **move up** $m\uparrow$ sur un arc e qui peut être utilisée pour déplacer l'arc e en direction des têtes, comme illustré par la figure 1 [2]. Le déplacement est contrôlé par une paire d'arguments $(\mathscr{A}, q)$ où $\mathscr{A}$ est un automate à états finis et q est un état de $\mathscr{A}$. L'automate représente toutes les transitions possibles à travers lesquelles l'arc peut se déplacer.

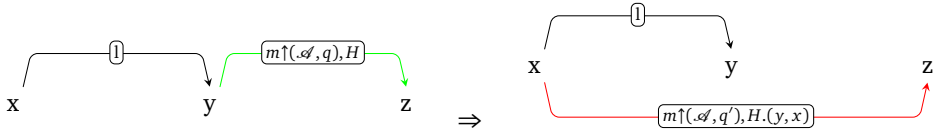

FIGURE 1: Contrainte **move up**

Prenons l'exemple de la figure 2, où l'antécédent du relatif *dont* est *Jean*. Ici, pour retrouver l'antécédent, il faut partir du pronom relatif, et remonter la série d'arcs : *de_obj* → *obj* → *obj* → *obj*, jusqu'à la source de l'arc *mod_rel*. De fait, le nombre d'arcs à remonter n'est pas borné. Il devient donc difficile d'écrire une règle simple pour ce cas de réécriture et il nous faut alors utiliser la contrainte *move up* pour assurer la remontée jusqu'à la source d'un *mod_rel*, assurant que l'on a retrouvé l'antécédent du relatif (l'arc rouge marque la transformation finale).

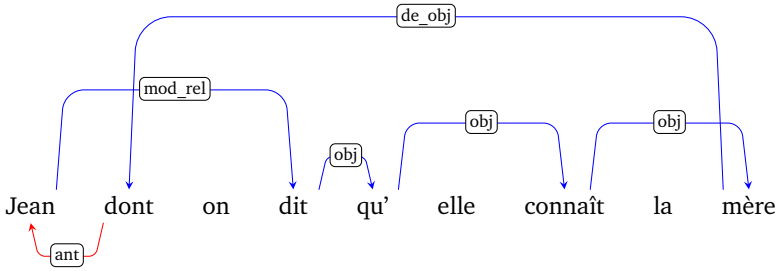

FIGURE 2: Arbre de dépendances pour *Jean dont on dit qu'elle connaît la mère.* En bleu, les arcs à suivre pour retrouver l'antécédent.

– Une contrainte **share up** $s\uparrow$ sur un arc $e = y \xrightarrow{l_e} z$ qui peut être utilisée pour dupliquer tous les arcs entrants $e' = x \xrightarrow{l} y$ de y comme arcs entrants de z (voir figure 3). Comme la contrainte $r\uparrow$, cette contrainte accepte un argument **L** qui restreint la duplication des arcs e' aux seuls arcs avec une étiquette $l \in \mathbf{L}$.

– Une contrainte **share down** $s\downarrow$ sur un arc $e = y \xrightarrow{l_e} z$ peut être utilisée pour dupliquer tous les arcs sortants de y comme arcs sortants de z, (voir figure 4). Cette contrainte accepte un argument **L** qui restreint la duplication des arcs e' avec une étiquette $l \in \mathbf{L}$. A noter, les arcs résultants ont une étiquette l^+, qui indique qu'ils ont été clonés.
Dans la figure 5, on remarque la présence de deux modaux, *vouloir* et *pouvoir*.

2. Où les arcs verts sont supprimés du graphe et les arcs rouges correspondent à la transformation finale.

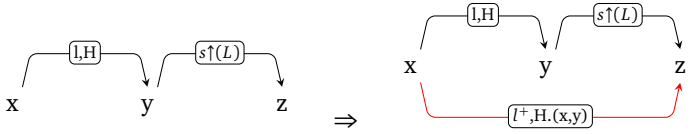

FIGURE 3: Contrainte **share up**

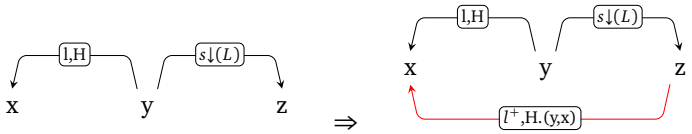

FIGURE 4: Contrainte **share down**

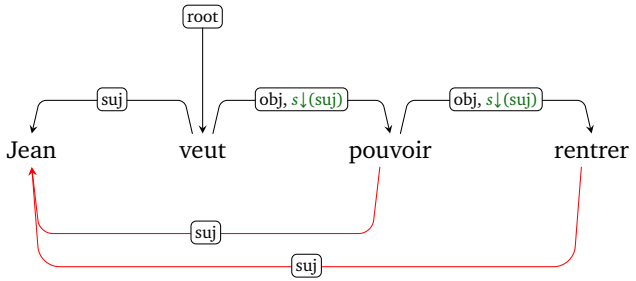

FIGURE 5: Arbre de dépendances pour *Jean veut pouvoir rentrer.*

Poser une contrainte de type *share_down* sur les deux arcs d'étiquette **obj**, conduit au partage du sujet, permettant ainsi de retrouver le sujet des verbes dans des phrases où il y en a plusieurs d'affilés.

Par ailleurs, les contraintes sont conçues pour permettre une interaction qui favorise la confluence et la terminaison du système. Prenons l'exemple *Jean qui m'aperçoit et m'appelle veut me parler*. On donne une représentation de cette phrase à la figure 6. Les arcs rouges décrivent les arcs finaux (après réécriture), l'arc portant la contrainte **move up**, qui est en vert foncé, est susceptible de bouger lors de la transformation et l'arc bleu est un arc temporaire détruit après transformation.

On voit qu'on utilise les trois contraintes précédemment explicitées. Deux contraintes **share down** permettent d'ajouter le sujet de *parler* et celui d'*appelle*. La contrainte **move up** ajoute l'antécédent du relatif « *qui* » ainsi qu'une contrainte **share up** qui permet d'ajouter le véritable sujet des verbes *aperçoit* et *appelle*.

La confluence est assurée par le fait que **move up** ne peut pas utiliser les arcs créés par **share down** ou **share up** et par le fait que les contraintes s'appliquent tant qu'elles le peuvent. Ainsi, on pourrait appliquer **move up**, puis **share down** et **share up** ou inverser **move up** et **share down** sans problème, le résultat serait toujours le même. Enfin, on voit que le nombre d'informations ajoutées est important au regard du nombre de contraintes posées sur les arcs.

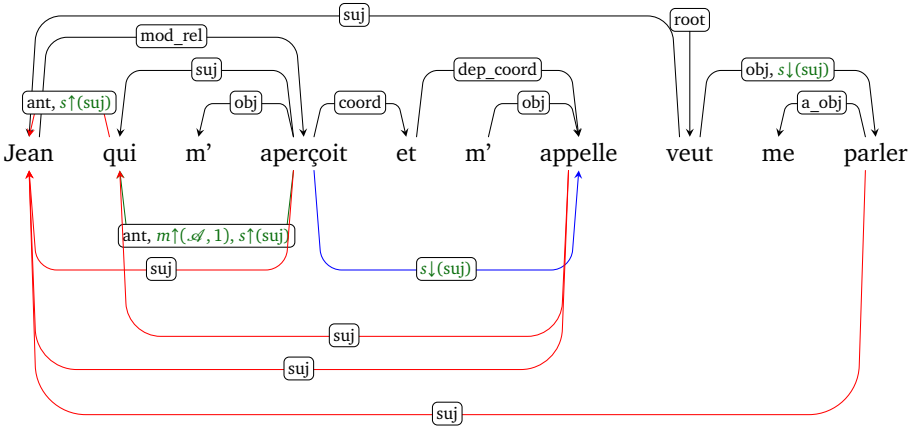

FIGURE 6: Exemple d'interaction entre contraintes

3 Illustration sur le contrôle du sujet et l'ellipse du sujet

Comme expliqué plus haut, nous avons utilisé notre système de réécriture pour ajouter des informations sur les cas de contrôles et d'ellipses du sujet.

3.1 Contrôle du sujet

Le contrôle du sujet est un problème lexical. En effet, la liste des verbes à contrôle est connue et peut être récupérée via un lexique tel que le LEFFF (Sagot, 2010). Cependant, il existe des exemples où le nombre de verbes à contrôle mis en relation peut être important, nous citerons le cas suivant issu du French Treebank : *Le Brésil veut pouvoir continuer à défricher l'Amazonie pour y installer ses colons affamés de terres cultivables* (phrase **5444** dans le FTB, section 1).

Nous sommes en présence d'un exemple qui demande de propager le sujet *Brésil* tout au long de la chaîne de modaux, jusqu'au verbe *défricher*. Ainsi, *Brésil* est sujet de *veut*, mais aussi de *pouvoir* et de *continuer à* et enfin de *défricher*.

3.2 Ellipse du sujet

Nous donnons à la figure 7 un exemple de représentation de l'ellipse sujet dans le French Treebank en dépendances.

Dans cet exemple, il faut partager le sujet *Jean* entre les deux conjoints de la coordination. Placer une contrainte *share down* entre *mange* et *boit* permet ce partage.

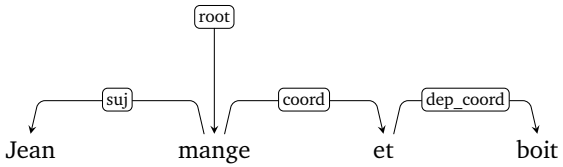

FIGURE 7: Représentation en dépendances de l'ellipse sujet selon le schéma d'annotation du FTB

4 Validation sur corpus

Pour valider notre travail, nous avons mis au point deux corpus, que nous décrivons en détail ci-dessous.

4.1 Les corpus

Pour nous permettre de nous évaluer, nous avons mis en place un sous-corpus par phénomène considéré. Ils sont tirés d'un ensemble de phrases du French Treebank (FTB) et du Sequoia-Bank (Candito et Seddah, 2012).

Ainsi, nous avons sélectionné un corpus de 405 phrases pour le contrôle obligatoire issues du SequoiaBank et du French Treebank que nous avons annotées à la main. En ce qui concerne la coordination à ellipse sujet, nous avons utilisé le corpus annoté de Bouchesèche (2009), auquel nous avons ajouté de nouvelles phrases pour un total de 120 phrases prises sur le French Treebank (section 2 et section 3 uniquement).

La répartition des phrases et du nombre total de dépendances pour chaque corpus est récapitulée dans le tableau 1.

PHÉNOMÈNE	DEV	TEST	TOTAL
Contrôle	155 (4223)	250 (6572)	405 (10795)
Ellipse du sujet	55 (1920)	65 (2423)	120 (4343)

TABLE 1: Nombre total de phrases (et de dépendances) dans les sous-corpus par phénomène étudié

Nos règles ont été mises au point sur les deux corpus de développement pour ensuite être testées sur les corpus de test. L'idée est de ne pas biaiser l'évaluation en les testant sur des phrases déjà connues.

Dans le tableau 2, nous indiquons le nombre de dépendances ajoutées lors de l'annotation manuelle.

Par ailleurs, on tient à attirer l'attention sur le fait que les deux phénomènes ne peuvent être comparés. En effet, la difficulté de la tâche est différente et le nombre de phrases dans chaque

PHÉNOMÈNE	DEV	TEST	TOTAL
Contrôle	345	578	923
Ellipse du sujet	116	129	145

TABLE 2: Nombre de dépendances ajoutées dans chaque sous-corpus

corpus n'est pas le même.

4.2 Protocole expérimental

Parseurs Pour tester la généricité de nos règles, nous avons analysé les corpus avec trois parseurs différents : le MaltParser (Nivre *et al.*, 2006), le MSTParser (McDonald *et al.*, 2005b) [3], et FrMG (Villemonte de La Clergerie, 2005). Les deux premiers sont des parseurs statistiques entraînés sur le French Treebank, le dernier est un parseur symbolique fondé sur une extension du formalisme des grammaires d'arbres adjoints (Joshi *et al.*, 1975; Joshi et Schabes, 1997). Nous comparons les sorties de FrMG préalablement transformées pour respecter le schéma d'annotation du FTB. En effet, les sorties natives de FrMG présentent un schéma d'annotation beaucoup plus profond que celui du FTB.

Corpus Le corpus qui contient les cas de contrôle a été constitué sur le French Treebank et le SequoiaBank en utilisant des phrases issues du corpus d'entraînement. Or, ce corpus a servi à générer les modèles des deux parseurs statistiques. De fait, nous les avons réentraînés avec les mêmes paramètres que Candito *et al.* (2010), en enlevant les phrases concernées du corpus d'entraînement. Par ailleurs, le corpus d'entraînement utilise des parties du discours prédites par un tagger, grâce à une méthode de rééchantillonnage (jackknifing [4]). Pour des questions de commodité, nous avons utilisé les prédictions de Morfette (Chrupala *et al.*, 2008).

Métriques utilisées Nous utilisons les métriques standards d'évaluation d'analyses syntaxiques en dépendances, à savoir le *Labeled Attachment Score* (LAS) qui correspond au pourcentage de dépendances correctes, étiquette incluse et le *Unlabeled Attachment Score* (UAS), qui correspond au pourcentage de dépendances correctes, sans tenir compte des étiquettes sur les arcs. Comme il est d'usage, les évaluations sont faites sans tenir compte des ponctuations. De plus, nous donnons les métriques de référence que sont le rappel, la précision et le F_1-score pour évaluer notre système. On a alors une bonne représentation de ce que le système est capable d'ajouter comme dépendances et on peut aussi mesurer sa capacité à ne pas surgénérer (ajouter plus de dépendances qu'il ne faudrait).

3. L'architecture utilisée pour le parsing de nos corpus est l'architecture du projet Bonsai (Candito *et al.*, 2010), qui a adapté les deux parseurs statistiques au Français.

4. L'objectif du jackknifing est d'obtenir un treebank avec des parties du discours prédites par un tagger qui n'a pas été entraîné sur les données qu'il étiquette. En d'autres termes, on souhaite que le treebank possède autant d'erreurs que ce que prédirait le tagger sur un corpus arboré non modifié. On fait donc de la validation croisée dix fois, i.e. le corpus d'entraînement est divisé en dix parties, on entraîne le tagger sur neuf parties et on étiquette la dixième.

4.3 Mise au point des règles

Comme nous le mentionnions ci-dessus, les règles ont été mises au point sur les corpus de développement afin de ne pas biaiser leur application sur les corpus de test. En effet, il est important de les appliquer sur des phrases inconnues.

Les règles ont été écrites en utilisant au maximum le mécanisme de propagation par contraintes, afin d'en tester la robustesse et la généricité. Nous avons regardé divers exemples dans le corpus de développement et essayé d'en retirer les caractéristiques générales. A chaque fois qu'une règle était créée, nous l'appliquions à notre corpus de développement pour voir ce qu'elle couvrait. Nous regardions alors les divergences et mettions au point une nouvelle règle ou modifions celles existantes. Le tableau 3 récapitule le nombre de règles sur chaque corpus.

CORPUS	NOMBRE DE RÈGLES
Contrôle	3
Ellipse du sujet	1
Total	**4**

TABLE 3: Nombre de règles pour chaque corpus

Règles pour le contrôle sujet Nous avons une règle qui gère le contrôle sujet avec des verbes transitifs, par exemple : *Il veut venir* où *vouloir* est le verbe à contrôle et *venir* le verbe contrôlé.

Nous produisons une représentation graphique de notre règle à la figure 8. On cherche un verbe de catégorie **V** (et qui par ailleurs est connu pour être un verbe à contrôle) qui gouverne n'importe quel verbe à l'infinitif (de catégorie **VINF**) avec une étiquette de type *obj* ou *ats* ou *dep* ou *aux_caus*. Si une telle configuration est trouvée, alors nous attachons une contrainte **share_down** sujet sur l'arc.

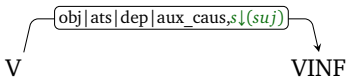

FIGURE 8: Règle pour le contrôle sujet avec des verbes transitifs

La deuxième règle gère le contrôle sujet des verbes intransitifs (*Il promet à Marie de venir*). La règle est illustrée à la figure 9. La configuration est sensiblement la même que pour 8, mais il faut prendre en compte l'ajout de la préposition. Si une telle configuration est trouvée, on ajoute un arc qui porte la contrainte **share_down** entre le verbe à contrôle et le verbe contrôlé.

Enfin, la troisième règle est particulière au verbe **venir** qui présente souvent une dépendance de type *mod* dans le FTB. Intégrer une telle dépendance à la première règle ferait baisser la précision du système.

Règle pour l'ellipse sujet La règle qui gère l'ellipse sujet est illustrée à la figure 10. Nous cherchons une configuration telle qu'un verbe possède un sujet et gouverne une conjonction

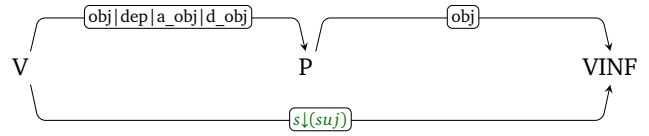

FIGURE 9: Règle pour le contrôle sujet avec des verbes intransitifs

de coordination qui elle-même gouverne un verbe (qui par ailleurs n'a pas de sujet). Si cette configuration existe, alors on crée un arc qui porte une contrainte **share_down** sujet entre les deux verbes. Les arcs gouvernés par le premier verbe seront partagés avec l'autre.

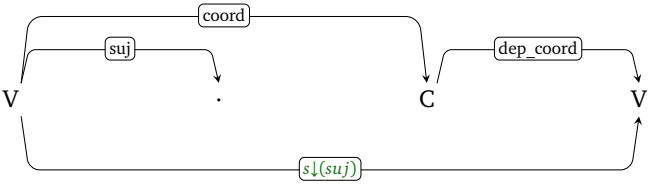

FIGURE 10: Règle pour l'ellipse sujet

L'idée qui se dessine avec la propagation de contraintes est que l'ordre d'application des règles est flexible donnant ainsi une plus grande liberté au système et à l'utilisateur qui écrit les règles. D'ailleurs, pour traiter les cas d'ellipse du sujet, il ne faudra que cette règle.

4.4 Résultats

Nous avons évalué deux choses différentes : la performance de nos règles et de notre système sur un corpus manuellement annoté et la généricité de nos règles sur des sorties d'analyseurs syntaxiques qui présentent, la plupart du temps, des erreurs par rapport à un corpus de référence.

Pour ce faire, nous avons appliqué nos règles sur notre corpus de référence sans les nouvelles annotations pour, ensuite, en évaluer l'impact. Les résultats sont reportés dans le tableau 4.

PHÉNOMÈNE	RAPPEL	PRÉCISION	F-SCORE
Contrôle	93,77	99,83	96,70
Ellipse du sujet	93,02	99,72	96,25

TABLE 4: Evaluation sur les corpus de test après applications des règles sur la référence (corpus gold)

Ensuite, pour mesurer la généricité de nos règles, il nous a paru intéressant de voir leur impact sur des sorties d'analyseurs syntaxiques. Nos deux corpus ont été parsés avec FrMG, mais aussi avec le MaltParser et le MSTParser. Nous avons évalué la performance des trois parseurs par rapport à notre corpus de référence. Les résultats sont donnés dans les tableaux 5 et 6.

	Dev		Test	
Parseur	*LAS*	*UAS*	*LAS*	*UAS*
FrMG	83,01	85,70	84,51	87,14
Malt	**86,50**	**89,41**	**87,55**	**90,05**
MST	84.34	87,48	85,82	88.65

Table 5: Contrôle : Evaluation des analyses de FrMG, de Malt et du MSTParser

	Dev		Test	
Parseur	*LAS*	*UAS*	*LAS*	*UAS*
FrMG	81,77	85,31	85,22	87,12
Malt	**85,31**	**87,35**	**86,05**	**88,07**
MST	83,54	86,77	83,45	85,89

Table 6: Ellipses : Evaluation des analyses de FrMG, de Malt et du MSTParser

Ensuite, nous avons appliqué notre système sur ces nouvelles analyses. Les résultats sont présentés dans les tableaux 7 et 8.

Parseur	LAS	UAS	Rappel	Précision	F-score
FrMG	84,24	86,77	85,81	99,64	92,21
Malt	87,07	89,50	83,91	99,71	91,13
MST	81,77	84,49	82,18	99,72	90,11
MST (Tagset réduit)	81,77	84,49	39,45	100,00	56,58

Table 7: Contrôle : Evaluation sur les sorties des trois parseurs après application des règles

Parseur	LAS	UAS	Rappel	Précision	F-score
FrMG	84,65	86,50	78,29	99,52	87,64
Malt	85,21	87,18	72,87	99,36	84,07
MST	82,56	84,97	68,21	99,39	80,91

Table 8: Ellipse : Evaluation sur les sorties des trois parseurs après application des règles

5 Discussion

Le tableau 4 donne les résultats lors de l'application des règles sur le corpus de référence. Les résultats montrent deux choses intéressantes :

1. La précision du système est excellente, ce qui signifie qu'il ne surgénère que très peu, n'ajoutant pas d'informations erronées au corpus.
2. Le rappel est certes moins bon, mais il reste tout de même haut : le système est capable d'ajouter un maximum de dépendances correctes.

En somme, sur le corpus de référence, les règles s'appliquent bien et ajoutent des dépendances qui nous permettent de nous approcher de la syntaxe profonde. De plus, on peut constater que pour des cas difficiles comme certaines coordinations elliptiques, le nombre d'erreurs n'est pas élevé.

Nous nous intéressons ensuite aux résultats sur les corpus analysés par les différents parseurs (tableaux 7 et 8) :

1. Nous obtenons de bons scores sur les analyses de FrMG, sans doute dû au fait que FrMG est un parseur produisant des analyses profondes, il est donc possible que les rattachements d'ellipses et de contrôle soient meilleurs.
2. Nous avons aussi de bons résultats sur les analyses de MaltParser pour les cas de contrôle. Cependant, les cas d'ellipses ne sont pas aussi concluants. En effet, Malt a des difficultés à retrouver les structures coordonnées, ce qui fait que les règles sont plus difficiles à appliquer ;
3. Concernant le MSTParser, dans une version préliminaire de ce papier, nous avions indiqué des résultats assez faibles concernant le rappel, indiquant que le parseur n'utilisait pas le même ensemble des parties du discours que les autres parseurs. En effet, suite à une erreur de paramétrage de notre part, le MSTParser a utilisé un ensemble plus restreint de parties du discours (V pour V, VINF, VPP, etc.). Les résultats présentés dans le tableau 7 corrigent ce problème. Néanmoins, par soucis d'exhausitivité nous rappelons à la ligne MSTParser (Tagset réduit) les anciens résultats obtenus.

L'utilisation du tagset réduit pour le MSTParser nous avait conduit à modifier nos règles dans ce sens, les rendant encore plus génériques. En effet, plus l'ensemble de parties du discours est réduit plus nous avons des chances d'appliquer nos règles sur un nombre plus important de structures, réduisant ainsi la précision du système. Nous avons donc évalué cet impact sur les différents corpus. Les résultats sont présentés dans le tableau 9.

Il est intéressant de constater que le fait de rendre les règles plus génériques entraîne une perte minime sur la référence, alors que nous avons un gain intéressant sur les analyses du MSTParser.

Au vue des résultats précédents, nous pouvons avancer que le système de propagation de contraintes fonctionne bien. Avec quelques contraintes, agissant sur des configurations simples, nous pouvons couvrir des phénomènes souvent difficiles à traiter, présentant des interactions. De plus, nous constatons que peu de règles suffisent à couvrir un phénomène et que les règles couvrant les cas particuliers sont évitées au maximum. En cela, le système reste générique. Par ailleurs, comme tout système à base de règles, il était important de voir à quel point les règles mises au point sur un corpus peuvent s'appliquer sur une version altérée de ce même corpus.

	LAS	UAS	RAPPEL	PRÉCISION	F-SCORE
FrMG	84,24	86,77	85,81	99,59	92,19
Malt	87,05	89.48	83,74	99,65	91,00
MST (Tagset réduit)	85,36	88,11	82,18	99,68	90,09
Référence	99,49	99,53	93,43	99,74	96,48

TABLE 9: CONTRÔLE : Evaluation après utilisation de règles plus génériques

Conclusion

A travers l'enrichissement d'un corpus arboré et de sortie d'analyseurs syntaxiques, nous avons montré qu'avec une approche générique de propagation par contraintes, il était possible d'avoir un système de réécriture de graphes plus facile à maintenir avec un nombre de règles restreint. Par ailleurs, grâce aux contraintes, il est possible d'exprimer des règles suffisamment génériques pour qu'elles puissent être utilisées sans modifications importantes sur des sorties prédites d'analyseurs syntaxiques.

Cela ouvre le champ à d'autres applications, parmi lesquelles nous pouvons citer la correction d'analyses syntaxiques afin d'améliorer les résultats d'un parseur. Dans la même mouvance, on peut tout autant essayer de réécrire des sorties d'analyseurs différents avec le même jeu de règles, pour ensuite comparer les graphes obtenus, afin de déduire le meilleur graphe à partir des différentes structures proposées. On pourrait améliorer les analyses syntaxiques au moyen de plusieurs parseurs et du système de réécriture qui guiderait les analyses.

Enfin, une autre piste à explorer, serait la transformation de schémas d'annotation dans le but d'évaluer les schémas entre eux. On sait que la comparaison de deux schémas d'annotations n'est pas chose aisée, ainsi utiliser un ensemble de règles, pour tenter de transformer un schéma en un autre permettrait une comparaison plus aisée des analyseurs syntaxiques entre eux.

Remerciements

Je souhaitais remercier Djamé Seddah et Éric de la Clergerie pour leurs conseils, leurs relectures attentives et leurs remarques lors de l'écriture de cet article.

Références

ABEILLÉ, A., CLÉMENT, L. et TOUSSENEL, F. (2003). Building a Treebank for French. *In Treebanks : Building and Using Parsed Corpora*, pages 165–188. Springer.

BONFANTE, G., GUILLAUME, B. et MOREY, M. (2011a). Modular graph rewriting to compute semantics. *In International Workshop on Computional Semantics 2011*.

BONFANTE, G., GUILLAUME, B., MOREY, M. et PERRIER, G. (2010). Réécriture de graphes de dépendances pour l'interface syntaxe-sémantique. *In TALN 2010*.

BONFANTE, G., GUILLAUME, B., MOREY, M. et PERRIER, G. (2011b). Enrichissement de structures en dépendances par réécriture de graphes. *In TALN 2011*.

BOUCHESÈCHE, L. (2009). Annotation semi-automatique des coordinations à ellipse sur corpus arboré. Mémoire de maîtrise, Univ. Paris Sorbonne 4.

CANDITO, M., NIVRE, J., DENIS, P. et HENESTROZA ANGUIANO, E. (2010). Benchmarking of Statistical Dependency Parsers for French. *In Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, Chine. Coling 2010.

CANDITO, M. et SEDDAH, D. (2012). Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. *In TALN 2012 - 19e conférence sur le Traitement Automatique des Langues Naturelles*, Grenoble, France.

CHRUPALA, G., DINU, G. et van GENABITH, J. (2008). Learning Morphology with Morfette. *In Proceedings of LREC 2008*.

FRUEWIRTH, T. et ABDENNADHER, S. (2003). *Essentials of Constraint Programming*. Springer-Verlag New York, Inc.

GEISS, R., BATZ, G. V., GRUND, D., HACK, S. et SZALKOWSKI, A. (2006). GrGen : A fast SPO-Based graph rewriting tool. *In International Conference on Graph Transformation*.

JOSHI, A. K., LEVY, L. et TAKAHASHI, M. (1975). Tree Adjunct Grammars. *Journal of Computer and System Science 10*, 10(1).

JOSHI, A. K. et SCHABES, Y. (1997). Tree-adjoining grammars. *Handbook of formal languages*, 3:69–124.

LÖWE, M., EHRIG, H., HECKEL, R., RIBEIRO, L., WAGNER, A. et CORRADINI, A. (1993). Algebraic approches to graph transformations. *Theoritical Computer Science*.

MARCHAND, J., GUILLAUME, B. et PERRIER, G. (2010). Motifs de graphe pour le calcul de dépendances syntaxiques complètes. *In TALN 2010*.

MCDONALD, R., CRAMMER, K. et PEREIRA, F. (2005a). Online large-margin training of dependency parsers. *In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

MCDONALD, R., PEREIRA, F., RIBAROV, K. et HAJIČ, J. (2005b). Non-projective dependency parsing using spanning tree algorithms. *In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

NIVRE, J., HALL, J. et NILSSON, J. (2006). MaltParser : A data-driven parser-generator for dependency parsing. *In Proc. of LREC-2006*.

RIBEYRE, C. (2012). Mise en place d'un systéme de réécriture de graphes appliqués á l'interface syntaxe-sémantique. Mémoire de master, Univ. Paris Diderot 7.

RIBEYRE, C., SEDDAH, D. et VILLEMONTE DE LA CLERGERIE, É. (2012). A Linguistically-motivated 2-stage Tree to Graph Transformation. *In* HAN, C.-H. et SATTA, G., éditeurs : *TAG+11 - The 11th International Workshop on Tree Adjoining Grammars and Related Formalisms - 2012*, Paris, France. INRIA.

SAGOT, B. (2010). The LEFFF, a freely available and large-coverage morphological and syntactic lexicon for french. *In Proceedings of LREC'10*, Valetta, Malta.

VILLEMONTE DE LA CLERGERIE, E. (2005). From metagrammars to factorized TAG/TIG parsers. *In Proceedings of IWPT'05 (poster)*, Vancouver, Canada.