

TALN 2007, Toulouse, 12–15 juin 2007

Une réalisateur de surface basé sur une grammaire réversible

Claire Gardent¹ Eric Kow²

(1) CNRS/LORIA, Nancy

(2) INRIA/LORIA, Nancy

Claire.Gardent@loria.fr, Eric.Kow@loria.fr

Résumé En génération, un réalisateur de surface a pour fonction de produire, à partir d'une représentation conceptuelle donnée, une phrase grammaticale. Les réalisateurs existants soit utilisent une grammaire réversible et des méthodes statistiques pour déterminer parmi l'ensemble des sorties produites la plus plausible ; soit utilisent des grammaires spécialisées pour la génération et des méthodes symboliques pour déterminer la paraphrase la plus appropriée à un contexte de génération donné. Dans cet article, nous présentons GENI, un réalisateur de surface basé sur une grammaire d'arbres adjoints pour le français qui réconcilie les deux approches en combinant une grammaire réversible avec une sélection symbolique des paraphrases.

Abstract In generation, a surface realiser takes as input a conceptual representation and outputs a grammatical sentence. Existing realisers fall into two camps. Either they are based on a reversible grammar and use statistical filtering to determine among the several outputs the most plausible one. Or they combine a grammar tailored for generation and a symbolic means of choosing the paraphrase most appropriate to a given generation context. In this paper, we present GENI, a surface realiser based on a Tree Adjoining Grammar for French which reconciles both approaches in that (i) the grammar used is réversible and (ii) paraphrase selection is based on symbolic means.

Mots-clefs : Réalisation de surface, Grammaire d'arbres adjoints, Réversibilité

Keywords: Surface realisation, Tree Adjoining Grammar, Reversibility

1 Introduction

En génération, le module de *réalisation de surface* a pour fonction de produire, à partir d'une représentation conceptuelle donnée, une phrase grammaticale. Par exemple, à partir de l'entrée donnée en (1a), un réalisateur de surface pourra produire l'une des variantes listée en (1b-1k).

- | | |
|---|---|
| <p>(1) a. <i>jean(j) aimer(e,j,m) marie(m)</i>
 b. Jean aime Marie
 c. Marie est aimée par Jean
 d. Marie est aimée de Jean
 e. C'est Marie que Jean aime
 f. C'est Jean qui aime Marie</p> | <p>g. C'est Marie qui est aimée par Jean
 h. C'est Marie qui est aimée de Jean
 i. Marie, Jean l'aime
 j. Marie, c'est Jean qui l'aime
 k. Jean, c'est Marie qu'il aime</p> |
|---|---|

Dans cet article, nous présentons un réalisateur de surface (GENI) qui combine une méthode symbolique de sélection de paraphrases avec une grammaire réversible. En outre, le réalisateur est paramétrable et peut être utilisé soit en mode déterministe (une seule solution produite), soit en mode non déterministe (toutes les paraphrases associées par la grammaire à la sémantique d'entrée sont produites).

L'utilisation de méthodes symboliques pour guider le choix de la paraphrase permet de prendre en compte les facteurs contextuels imposés par un système de génération : la réalisation produite est une réalisation *appropriée* pour un contexte donné plutôt que la réalisation la plus fréquente dans l'usage général représenté par un corpus d'apprentissage.

L'utilisation d'une grammaire réversible a plusieurs avantages.

Premièrement, elle permet d'utiliser le même lexique et la même grammaire pour l'analyse et pour la réalisation. Etant donnée la difficulté de développer de telles ressources, la question de la réutilisabilité est un point non trivial.

Deuxièmement, comme l'illustre la Redwood Lingo Treebank (Velldal & Oepen, 2006), la réversibilité permet de créer rapidement de très grandes suites de tests pour la réalisation : il suffit pour ce faire d'analyser des phrases, de sélectionner parmi les sorties produites l'analyse correspondant à l'interprétation de l'entrée et d'utiliser la représentation sémantique associée comme entrée pour le réalisateur. Par comparaison, les plus grandes suites de tests distribuées actuellement avec les réalisateurs existants soit sont de taille restreinte (500 phrases distinctes pour SURGE, 210 pour KPML), soit exigent de développer un module de transformation permettant de créer à partir d'un corpus arboré un format d'entrée adapté à la réalisation (Callaway, 2003).

Troisièmement, la réversibilité permet de mieux mesurer la couverture et le pouvoir paraphrastique du réalisateur. La couverture est testée par une analyse doublée d'une phase de réalisation (la phrase d'entrée peut-elle être à la fois analysée et réalisée par le système?). Le pouvoir paraphrastique pourra être mesuré en utilisant le réalisateur en mode non déterministe (toutes les paraphrases possibles sont produites) et en identifiant les sorties correctes produites par le réalisateur pour une entrée donnée.

Quatrièmement, une grammaire réversible peut être utilisée à la fois pour la réalisation et pour son inverse à savoir, la construction sémantique (i.e., la construction pour une phrase de sa ou ses représentations sémantique(s)). Si, comme nous cherchons à le garantir dans GENI, le réalisateur a un bon pouvoir paraphrastique, cela a pour effet que la grammaire peut être utilisée à la fois pour générer et pour détecter les paraphrases.

L'article est structuré de la façon suivante. Nous commençons par présenter la grammaire (section 2) et l'algorithme de base (section 3.1). Ce premier algorithme est un algorithme non déterministe qui produit l'ensemble des paraphrases associées par la grammaire à une représentation sémantique donnée. Nous montrons ensuite comment une méthode de paramétrisation des entrées permet de restreindre la réalisation aux seules paraphrases respectant les critères syntaxico-sémantiques spécifiés par les paramètres donnés en entrée (section 3.2). La section 4 présente des résultats chiffrés donnant une indication de la couverture et du pouvoir paraphrastique de GENI et la section 5 compare l'approche proposée avec les travaux connexes. La section 6 conclut avec des pointeurs pour des recherches futures.

2 La grammaire

Formalisme. La grammaire utilisée est une grammaire d'arbres adjoints lexicalisée basée sur l'unification (FLTAG, (Vijay-Shanker & Joshi, 1988)). Une FLTAG comprend un ensemble d'arbres élémentaires et deux opérations permettant de combiner ces arbres entre eux, l'opération de substitution et l'opération d'adjonction. Les arbres résultant d'une de ces opérations sont appelés «arbres dérivés».

Les arbres élémentaires sont lexicalisés, c'est-à-dire qu'ils sont explicitement associés avec un lemme ou une forme fléchie. Leurs noeuds sont étiquetés par deux structures de traits appelées TOP et BOTTOM. Un arbre élémentaire est soit initial, soit auxiliaire. Un arbre initial est un arbre dont les noeuds feuilles sont soit des noeuds terminaux, soit des noeuds dit de substitution (marqués par \downarrow). Un arbre auxiliaire est un arbre dont l'un des noeuds feuilles est un noeud «pied» (marqué par \star) étiqueté par la même catégorie que le noeud racine.

L'opération de substitution permet d'insérer un arbre élémentaire ou dérivé τ_δ dans un arbre initial τ_α : le noeud racine de τ_δ est alors identifié avec un noeud de substitution dans τ_α et les traits TOP sont unifiés ($Top_{\tau_\alpha} = Top_{\tau_\delta}$). L'opération d'adjonction permet d'insérer un arbre auxiliaire τ_β dans un arbre quelconque τ_α à un noeud n : les traits TOP_n et $BOTTOM_n$ du noeud n où se fait l'adjonction sont alors unifiés avec les traits TOP du noeud racine de l'arbre auxiliaire et les traits BOTTOM de son noeud pied respectivement ($Top_n = Top_{Root_{\tau_\beta}}$ et $Bottom_n = Bottom_{Foot_{\tau_\beta}}$). En fin de dérivation, les traits TOP et BOTTOM de chaque noeud de l'arbre dérivé produit sont unifiés.

Grammaire et méta-grammaire. La grammaire TAG utilisée est une grammaire produite par compilation à partir d'une spécification plus abstraite appelée, *metagrammaire*. Dans cette méta-grammaire, un arbre élémentaire est défini par la combinaison d'un ou de plusieurs fragments d'arbres et chaque fragment d'arbre encapsule une caractéristique linguistique spécifique. Par exemple, tout arbre verbal dont le sujet est un sujet nominal canonique fera intervenir le fragment d'arbre SUJETCANONIQUE. Plus généralement, chaque arbre élémentaire est associé par le processus de compilation à un *profil* listant l'ensemble des noms de fragments d'arbres ayant participé à sa construction. Comme l'illustre le profil de l'arbre donné en Figure 1, le profil de chaque arbre donne ainsi des informations sur ses caractéristiques linguistiques. Or les réalisateurs TAG utilisent souvent ce type d'information pour guider la réalisation (cf. (Yang *et al.*, 1991; Danlos, 1998)) mais l'association est faite de manière manuelle ; Dans GENI, ces informations sont automatiquement associés avec chaque arbre élémentaire par le compilateur de méta-grammaire.

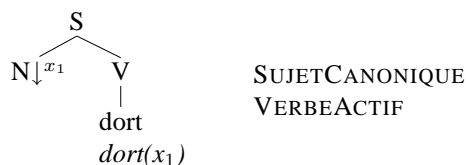


FIG. 1 – Profil d’un arbre

Interface Syntaxe/Sémantique. Dans la grammaire d’arbres adjoints utilisée, le lien entre structure syntaxique et représentation sémantique se fait de la façon illustrée par la figure 2. Chaque arbre élémentaire est associé avec une représentation sémantique où les arguments manquants sont des variables d’unification. Ces variables apparaissent en outre sur certains noeuds de l’arbre et sont instantiées par le biais des substitutions et des adjonctions¹. Ainsi, dans la dérivation de *Jean court souvent* illustré ci-dessous, j unifie avec s et r avec x si bien que la représentation sémantique finale est $nom(j,jean), courir(r,j), souvent(r)$.

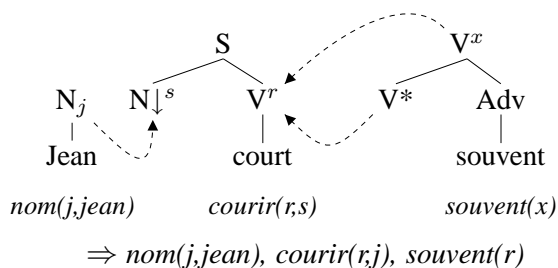


FIG. 2 – “Jean court souvent”

Couverture. La grammaire (Crabbé & Duchier, 2004) couvre la grammaire noyau du français décrite dans (Abeillé, 2002) c’est-à-dire, l’ensemble des cadres syntaxiques de base qui y sont listés et pour chacun de ces cadres, l’ensemble des redistributions (actif, passif, moyen, neutre, réflexivisation, impersonnel, passif-impersonnel) et des réalisations d’arguments permises (cliticisation, extraction, omission, variations d’ordre).

3 Le réalisateur GENI

Nous commençons par décrire l’algorithme de base utilisé par GENI. Cet algorithme est un algorithme non déterministe permettant de générer l’ensemble des paraphrases associées par la grammaire à une entrée sémantique telle que celle donnée en (1a). Nous montrons ensuite comment cet algorithme peut être modifié pour assurer le déterminisme et générer parmi l’ensemble des paraphrases possibles, la paraphrase respectant les contraintes placées sur une verbalisation par un contexte de génération donné.

3.1 Générer toutes les paraphrases

L’algorithme de base est un algorithme tabulaire (Kay, 1996) ascendant optimisé pour les grammaires d’arbres adjoints. Une spécification détaillée de l’algorithme et des optimisations dé-

¹Pour plus de détails sur le calcul sémantique utilisé, cf. (Gardent & Kallmeyer, 2003).

ployées est donnée dans (Gardent & Kow, 2005). Par manque de place, nous nous contentons ici d'illustrer son fonctionnement par un exemple.

Supposons que la sémantique donnée en entrée soit $\text{courir}(r, j), \text{nom}(j, \text{jean}), \text{souvent}(r)$. L'algorithme procède de la façon suivante. Dans un premier temps (**phase de sélection lexicale**), les arbres élémentaires dont la sémantique subsume une partie de l'entrée sont sélectionnés. Pour notre exemple, les arbres sélectionnés seront (entre autre) les arbres de *Jean*, *court* et *souvent* (cf. Figure 2). La deuxième étape (**phase de substitution**) consiste à explorer systématiquement les possibilités de combinaisons par substitution. Pour l'exemple considéré, cette exploration permettra de substituer l'arbre pour *Jean* dans l'arbre pour *court* (cf. Figure 2). La troisième étape (**phase d'adjonction**) permet de combiner les arbres produits par adjonction. C'est à ce stade que l'arbre pour *souvent* sera adjoint à l'arbre dérivé pour *Jean court*. En dernier ressort (**phase d'extraction**), les chaînes étiquettant les items couvrant la sémantique donnée en entrée sont produites en l'occurrence : *Jean court souvent*.

3.2 Générer une seule paraphrase

L'algorithme présenté dans la section précédente est non déterministe et produit, pour une entrée sémantique donnée, l'ensemble des paraphrases associées par la grammaire à cette entrée. L'entrée (1a) par exemple, permet la réalisation de l'ensemble des paraphrases listées en (1b-1k).

Pour une utilisation au sein d'un système de génération, il est nécessaire de pouvoir contraindre l'algorithme de façon à pouvoir sélectionner, parmi l'ensemble des paraphrases possibles, la paraphrase appropriée au contexte considéré. Nous montrons maintenant comment l'entrée sémantique de GENI peut être enrichie pour guider l'algorithme dans ses choix et assurer le déterminisme.

Au plus une verbalisation. Dans l'algorithme présenté en section 3, le non déterminisme provient principalement² de l'ambiguïté lexicale : pour chaque littéral l dans l'entrée, il y a généralement plus d'un arbre élémentaire sélectionné par la phase de sélection lexicale. Ainsi pour chaque entrée sémantique, la sortie de GENI est donnée par l'ensemble des combinaisons d'arbres élémentaires couvrant la sémantique d'entrée et dont la combinaison par substitution ou adjonction est permise par la grammaire.

Pour permettre le déterminisme, nous associons à chaque littéral un *identifiant d'arbre*. Comme nous l'avons vu en section 2, la grammaire utilisée est produite par un processus de compilation qui associe à chaque arbre élémentaire un profil résumant ses caractéristiques linguistiques. Parce que ce profil recense l'ensemble des fragments d'arbres utilisés pour construire un arbre donné et parce que chaque arbre diffère d'un autre par au moins un fragment d'arbre dans son profil, le profil de chaque arbre élémentaire est un identifiant pour cet arbre³. Nous utilisons cette caractéristique de la grammaire pour guider la réalisation et assurer son déterminisme de

²Une seconde source de non déterminisme provient des modificateurs qui peuvent souvent s'adjoindre dans des ordres différents (*l'homme jeune et grand, l'homme grand et jeune*). Ce type de non déterminisme est traité dans la phase d'adjonction en regroupant les modificateurs d'une même entité et en imposant un ordre unique d'adjonction en cas d'ambiguïté.

³Il s'agit là d'une simplification. Dans certains cas (peu nombreux), deux arbres élémentaires distincts ont le même profil. Nous revenons sur ce point en section 4.

la façon suivante :

1. Chaque littéral l_i dans l'entrée est associée avec un identifiant d'arbre A_i . Cet identifiant est un profil simplifié ne prenant en compte que les informations utiles pour la génération et préservant l'unicité de l'arbre identifié.
2. Pendant la réalisation, pour chaque paire $l_i : A_i$ dans l'entrée enrichie, la sélection lexicale est restreinte aux arbres dont la sémantique subsume l_i et dont le profil simplifié est A_i .

Puisque chaque littéral est associé avec un identifiant d'arbre et chaque identifiant d'arbre identifie un arbre unique, le réalisateur produira *au plus* une phrase. Les exemples (2a-2c) illustrent le type de contraintes mises en jeu par GENI.

- (2) a. $l_j : \text{jean}(j)/\text{NomPropre}$ $l_a : \text{aimer}(e,j,m)/[\text{SujetCanoniqueNominal}, \text{VerbeActif}, \text{ObjetCanoniqueNominal}]$ $l_m : \text{marie}(m)/\text{NomPropre}$
 Jean aime Marie
~~Marie est aimée de Jean, C'est Jean qui aime Marie, etc.~~
- b. $l_c : \text{le}(c)/\text{Det}$ $l_c : \text{chien}(c)/\text{NomCommun}$ $l_d : \text{dormir}(e1,c)/\text{SujetRelatif}$
 $l_r : \text{ronfler}(e2,c)/\text{SujetCanoniqueNominal}$
 Le chien qui dort ronfle
~~Le chien qui ronfle dort~~
- c. $l_j : \text{jean}(j)/\text{ProperName}$ $l_p : \text{promettre}(e1,j,m,e2)/[\text{SujetCanoniqueNominal}, \text{VerbeActif}, \text{ObjetCompletive}]$ $l_m : \text{marie}(m)/\text{ProperName}$ $l_{e2} : \text{partir}(e2,j)/\text{VerbeInfinitif}$
 Jean promet à Marie de partir
~~Jean promet à Marie qu'il partira~~

Au moins une verbalisation. Si une entrée enrichie permet de limiter la réalisation à une verbalisation maximum, elle ne garantit pas une solution au sens où la combinaison d'identifiants d'arbres choisie peut ne pas être réalisable. Mais comment vérifier la satisfiabilité d'une entrée sans pour autant tenter de la réaliser ? Les systèmes existants donnent trois grands types de réponses à cette question.

Une première possibilité est de construire simultanément entrée enrichie et verbalisation. C'est le cas en particulier, des réalisateurs basés sur les grammaires systémiques (KPML(Matthiessen & Bateman, 1991)) où la réalisation coïncide avec la traversée d'un réseau systémique permettant d'associer à un contenu sémantique (dimension idéationnelle dans la terminologie systémique), une description syntaxique et fonctionnelle.

Une deuxième possibilité consiste à vérifier la validité de l'entrée sur un critère partiel de bonne formation. Par exemple, REALPRO (Lavoie & Rambow, 1997) prend pour entrée un structure syntaxique profonde de la théorie Sens-Texte (Mel'čuk & Žolkovskij, 1970) et SURGE (Elhadad & Robin, 1999) une description fonctionnelle des grammaires fonctionnelles d'unification. Dans les deux cas, l'entrée n'est pas nécessairement satisfiable puisqu'elle doit être validée par l'ensemble de la théorie. Dans la théorie sens-texte, la structure syntaxique profonde doit, pour être réalisable, pouvoir être projetée successivement en une structure syntactique de surface, une structure morphologique et une structure phonétique. Dans SURGE, l'entrée n'est réalisable que si elle permet une traversée complète de la grammaire du noeud racine aux noeuds lexicaux instanciant la sémantique d'entrée.

Une troisième possibilité consiste à procéder de façon incrémentale et à faire des choix locaux (algorithme gourmand) guidés par les contraintes introduites par les choix faits. C'est le cas

en particulier de l'algorithme semi-récurif basé sur les grammaires d'arbres adjoints décrit dans (Danlos, 1998) : à chaque étape de la réalisation, un arbre unique est sélectionné et les contraintes introduites par cet arbre sont utilisées pour contraindre le choix des arbres restant à sélectionner. En cas d'échec, l'algorithme fait un retour arrière.

Nous adoptons une stratégie mixte où la technique dite de *filtrage par polarités* est utilisée pour :

1. effectuer une vérification de la validité de l'entrée enrichie
2. proposer une alternative en cas d'échec de cette vérification
3. proposer une entrée enrichie par défaut en cas d'échec de réalisation

Le filtrage par polarités s'inspire des travaux de (Bonfante *et al.*, 2003; Koller & Striegnitz, 2002) et vise à détecter les combinaisons d'arbres élémentaires où besoins et ressources syntactico-sémantiques échouent à s'annuler. Plus spécifiquement, l'idée est de vérifier si pour une entrée donnée, l'ensemble d'arbres sélectionnés est tel que *chaque noeud de substitution et chaque noeud pied peut être associé avec exactement un arbre de la catégorie syntaxique et de l'index sémantique approprié*.⁴

Nous utilisons le filtrage par polarité pour filtrer les entrées enrichies non satisfiables (i.e., des entrées dont la polarité est non nulle) mais également pour proposer une entrée alternative en cas de détection d'entrée non satisfiable. Dans ce cas, l'entrée enrichie est dépouillée de ses identifiants d'arbre et l'algorithme non déterministe est utilisé avec additionnellement une étape de filtrage par polarité (introduite entre la phase de sélection lexicale et celle de substitution). Les combinaisons d'arbres à polarités neutres (i.e., les combinaisons d'arbres où besoins et ressources syntactico-sémantiques s'annulent) sont ensuite comparées à l'entrée enrichie initiale et la combinaison la plus similaire à cette entrée est alors proposée en alternative. La similarité entre deux combinaisons d'arbres est mesurée par le nombre d'identifiant communs entre ces deux combinaisons : plus le nombre d'identifiants commun aux deux sélections est grand, plus les sélections sont similaires. Si plusieurs sélections sont également similaires, un choix non déterministe est fait.

Enfin il est malgré tout possible qu'un ensemble d'arbres à polarité neutre soit non satisfiable⁵. Dans ce cas, l'algorithme utilise un enrichissement de la sémantique d'entrée par défaut qui permet de générer une verbalisation «canonique» de cette sémantique.

4 Evaluation

Afin d'évaluer d'une part, le pouvoir paraphrastique du réalisateur et d'autre part, l'impact des annotations de contrôle sur le non-déterminisme, nous avons utilisé une suite de tests graduée. Cette suite a été construite en (i) analysant un ensemble de phrases et (ii) sélectionnant pour chaque phrase la représentation sémantique correcte⁶. Le résultat est une suite de 80 représen-

⁴Les restrictions d'espace nous empêchent d'explicitier ici la méthode de filtrage par polarité. Celle-ci est cependant détaillée dans (Gardent & Kow, 2005).

⁵Le filtrage par polarité permet de détecter la non-satisfiabilité d'une combinaison d'arbres pas d'assurer sa satisfiabilité.

⁶L'analyseur peut donner plusieurs analyses et donc souvent plusieurs représentations sémantiques dont certaines représentent correctement le sens de la phrase analysée, d'autres non.

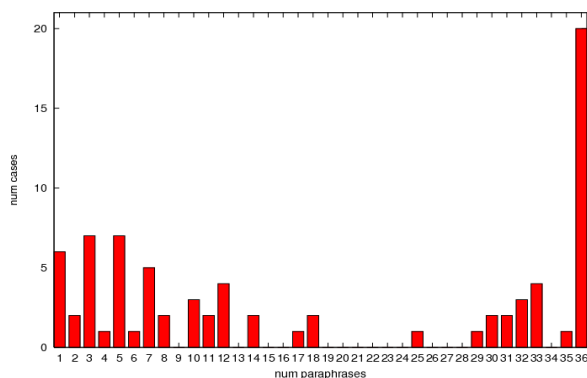


FIG. 3 – Distribution du taux de variation paraphrastique

tations sémantiques choisies pour illustrer les différents types de paraphrases grammaticales décrites par la grammaire utilisée c-à-d,

- les variations grammaticales dans la réalisation des arguments (clivés, cliticisation, extraction, inversion du sujet, etc.) et la forme du verbe (passive/active, impersonnel, etc.)
- les variations dans la réalisation des modificateurs (anté- vs post-posés, adjectif vs subordonnée relative, épithète vs. attribut, etc.)
- les variations permises par une équivalence morpho-dérivationnelle (Ex. arrivée/arriver)

Les 80 cas sélectionnés donnent lieu à la génération par GENI de 1 528 phrases distinctes soit un taux de paraphrases moyen par entrée de 18 avec une variation allant de 1 à plus de 50 paraphrases par représentation sémantique. La Figure 3 donne une description plus détaillée de la distribution du taux de variation paraphrastique. Plus généralement, 42% des phrases avec un verbe fini ont une à 3 paraphrases (cas des verbes intransitifs), 44% 4 à 28 paraphrases (verbes prenant deux arguments) et 13% acceptent plus de 30 paraphrases (verbes à trois arguments). Pour les phrases contenant deux verbes finis, le ratio est de 5% des cas ayant 1 à 3 paraphrases, 36% des cas ayant entre 4 et 14 paraphrases et 59% plus de 14 paraphrases. Enfin les phrases contenant plus de 3 verbes finis acceptent toutes plus de 20 paraphrases.

Afin de vérifier que l'utilisation des profils suffit à assurer le déterminisme (cf. section 3.2), nous avons calculé le nombre de cas où deux paraphrases d'un même contenu partagent le même profil. Pour ce faire, nous avons étiqueté de façon automatique les 1 528 paraphrases produites par GENI à partir de la suite de test, avec leur profils (le profil d'une paraphrase est l'ensemble des profils associés aux arbres élémentaires utilisés pour construire l'arbre dérivé de cette paraphrase). Nous avons ensuite comparé, pour chaque entrée de la suite de tests, les profils de toutes les paires de paraphrases correspondantes et compté le nombre de fois où une paire de paraphrases partage le même profil.

Cette manipulation montre que pour les 1 528 paraphrases considérées, le profil échoue à refléter la différence entre deux paraphrases dans moins de 1% des cas. L'analyse des données fautives révèle que les cas posant problème sont les paires paraphrastiques impliquant uniquement (i) une variation d'ordre des arguments (Ex. *Jean donne une pomme à Marie / Jean donne à Marie une pomme*) ou (2) une variation de position pour un modificateur (Ex. *Jean donne ce soir une pomme à Marie / Jean donne une pomme ce soir à Marie / Jean donne une pomme à Marie ce soir*). Le premier cas peut être résolu en modifiant la grammaire de façon à expliciter la différence dans le profil des arbres élémentaires correspondant, le second en imposant un ordre

canonique sur l'adjonction des modificateurs.

5 Discussion et comparaison avec travaux connexes

GENI diffère des réalisateurs existants en ce qu'il combine l'utilisation d'une grammaire réversible avec un mécanisme symbolique de sélection des paraphrases.

Ainsi, les réalisateurs utilisant une grammaire réversible utilisent généralement des méthodes statistiques pour choisir parmi l'ensemble des paraphrases associées à un sens par la grammaire, la paraphrase la plus fréquente plutôt que la paraphrase appropriée à un contexte de génération donné. Ils sont de ce fait mal adaptés à une utilisation dans des scénarios de génération exigeant un bon pouvoir paraphrastique et sont plutôt mis en jeu soit dans des systèmes de traduction (Carroll & Oepen, 2005) soit dans des scénarios de génération où les textes à produire sont fortement stéréotypés et où la paraphrase la plus fréquente est aussi la paraphrase la plus appropriée dans une majorité des cas (White, 2004).

Par ailleurs, les réalisateurs utilisant un mécanisme symbolique de sélection des paraphrases (Danlos, 1998; Matthiessen & Bateman, 1991; Elhadad & Robin, 1999; Lavoie & Rambow, 1997) mettent habituellement en jeu des grammaires qui ne sont pas réversibles c'est-à-dire des grammaires qui ne peuvent pas être exploitées à la fois pour associer un sens à un texte (analyse) et pour verbaliser un sens donné (réalisation). Or comme nous l'avons mentionné dans l'introduction, l'utilisation de la grammaire en analyse comme en réalisation est un point important. Outre qu'elle permet de minimiser l'effort de développement sur la grammaire (chaque modification apportée à la grammaire ou au lexique bénéficie à la fois à l'analyse et à la génération), cette dualité permet un meilleur contrôle de la qualité et de la couverture de la grammaire puisque l'analyseur permet de détecter la sous-génération (i.e., la non génération de phrases grammaticales) et le réalisateur la sur-génération (i.e., la génération de phrases agrammaticales). Enfin, l'utilisation en mode double de la grammaire permet de produire automatiquement des quantités arbitraires d'entrées pour le réalisateur et ainsi de mieux tester son pouvoir paraphrastique. Par comparaison, l'entrée des réalisateurs non réversibles est généralement produite soit par un système de génération, soit manuellement ce qui rend difficile une évaluation précise de leur couverture et pouvoir paraphrastique.

6 Conclusion

En résumé, GENI est un réalisateur de surface basé sur une grammaire d'arbres adjoints pour le français qui présente les propriétés suivantes :

- GENI peut être utilisé en mode déterministe (une paraphrase) ou non-déterministe (toutes les paraphrases)
- GENI utilise une grammaire et un lexique réversible
- GENI utilise une grammaire qui associe aux paraphrases grammaticales une même sémantique – ceci permet, grâce à la réversibilité, à la fois un bon pouvoir paraphrastique et la détection de paraphrases
- GENI est disponible en libre source (<http://trac.loria.fr/~geni>).

Outre les phénomènes décrits en section 4, la grammaire utilisée par GENI est essentiellement la grammaire spécifiée par (Crabbé, 2005) augmentée avec une dimension sémantique (Gardent,

2006). Elle couvre donc l'essentiel de la TSNLP et des phénomènes décrits dans (Abeillé, 2002). Le travail futur inclut (i) un passage à grande échelle par l'extension de la grammaire et du lexique et (ii) l'utilisation du réalisateur pour réduire la sur-génération.

Références

- ABEILLÉ A. (2002). *Une grammaire électronique du français*. CNRS Editions.
- BONFANTE G., GUILLAUME B. & PERRIER G. (2003). Analyse syntaxique électrostatique. *Évolutions en analyse syntaxique, Revue TAL (Traitement Automatique des Langues)*, **44**(3).
- CALLAWAY C. B. (2003). Evaluating coverage for large symbolic NLG grammars. In *18th IJCAI*.
- CARROLL J. & OEPEN S. (2005). High efficiency realization for a wide-coverage unification grammar. *2nd IJCNLP*.
- CRABBÉ B. (2005). *Représentation informatique de grammaires fortement lexicalisées*. PhD thesis, Université Henri Poincaré, Nancy.
- CRABBÉ B. & DUCHIER D. (2004). Metagrammar redux. In *CSLP 2004, Copenhagen*.
- DANLOS L. (1998). G-TAG : un formalisme lexicalisé pour la génération de textes inspiré de TAG. *Traitement Automatique des Langues - T.A.L.*, **2**.
- ELHADAD M. & ROBIN J. (1999). SURGE : a comprehensive plug-in syntactic realization component for text generation. *Computational Linguistics*.
- GARDENT C. (2006). Intégration d'une dimension sémantique dans les grammaires d'arbres adjoints. In *TALN 2006*.
- GARDENT C. & KALLMEYER L. (2003). Semantic construction in FTAG. In *10th EACL, Budapest*.
- GARDENT C. & KOW E. (2005). Generating and selecting grammatical paraphrases. *ENLG*.
- KAY M. (1996). Chart Generation. In *34th ACL*, p. 200–204, Santa Cruz, California.
- KOLLER A. & STRIEGNITZ K. (2002). Generation as dependency parsing. In *40th ACL, Philadelphia*.
- LAVOIE B. & RAMBOW O. (1997). RealPro—a fast, portable sentence realizer. *ANLP'97*.
- MATTHIESSEN C. & BATEMAN J. (1991). *Text generation and systemic-functional linguistics : experiences from English and Japanese*. Frances Pinter Publishers and St. Martin's Press.
- MEL'ČUK I. & ŽOLKOVSKIJ A. (1970). Towards a functioning meaning-text model of language. *Linguistics*, **57**, 10–47.
- VELLDAL E. & OEPEN S. (2006). Statistical ranking in tactical generation. In *EMNLP, Sydney*.
- VIJAY-SHANKER K. & JOSHI A. (1988). Feature Structures Based Tree Adjoining Grammars. *12th Computational linguistics*, **55**, v2.
- WHITE M. (2004). Reining in CCG chart realization. In *INLG*, p. 182–191.
- YANG G., MCKOY K. & VIJAY-SHANKER K. (1991). From functional specification to syntactic structure. *Computational Intelligence*, **7**, 207–219.