TALN 2005, Dourdan, 6–10 juin 2005

Comment obtenir plus des Méta-Grammaires

François Thomasset, Eric Villemonte de la Clergerie
ATOLL - INRIA
Domaine de Voluceau
Rocquencourt, B.P. 105, 78153 Le Chesnay (France)
{Francois.Thomasset,Eric.De_La_Clergerie}@inria.fr

Mots-clefs : Méta-grammaires, Analyse Syntaxique, TAG, TIG

Keywords: Meta-grammars, Parsing, TAG, TIG

Résumé Cet article présente un environnement de développement pour les méta-grammaires (MG), utilisé pour concevoir rapidement une grammaire d'arbres adjoints (TAG) du français à large couverture et néanmoins très compacte, grâce à des factorisations d'arbres. Exploitant les fonctionnalités fournies par le système DYALOG, cette grammaire a permis de construire un analyseur syntaxique hybride TAG/TIG utilisé dans le cadre de la campagne d'évaluation syntaxique EASY.

Abstract This paper presents a development environment for Meta-Grammars (MG), used to design, in a short period, a wide coverage but still very compact Tree Adjoining Grammar (TAG) for French, thanks to tree factorizations. Exploiting the functionalities provided by DYALOG system, an hybrid TAG/TIG parser was compiled from the grammar and used for the EASY parsing evaluation campaign.

1 Introduction

Les méta-grammaires (MG) (Candito, 1999) renouvellent les méthodes de conception des grammaires, en introduisant un niveau plus abstrait de description à l'aide de contraintes élémentaires, regroupées en classes relativement simples, elles-mêmes insérées dans une hiérarchie multiple d'héritage. Une phase de compilation permet ensuite de croiser ces classes et d'utiliser les contraintes pour dériver des structures grammaticales pour un formalisme cible comme les grammaires d'arbres adjoints (TAG) ou les grammaires fonctionnelles lexicales (LFG) (Gaiffe *et al.*, 2003; Clément & Kinyon, 2003). Les descriptions deviennent plus modulaires et permettent la factorisation d'ensembles de contraintes communs à plusieurs phénomènes syntaxiques (comme des règles d'accord). L'héritage permet d'affiner progressivement la description d'un phénomène, par exemple pour la structure verbale. Il rend aussi raisonnable l'espoir qu'une partie de l'organisation en classes ainsi qu'une partie du contenu des classes puissent être conservées d'une langue à une autre et d'un formalisme cible à un autre.

Ces raisons nous ont conduit à choisir les méta-grammaires pour concevoir rapidement un analyseur syntaxique hybride TAG/TIG du français à large couverture, analyseur qui a finalement pu être déployé dans le cadre de la campagne EASY d'évaluation d'analyseurs syntaxiques. Néanmoins, nos premières tentatives ont montré certaines limites dans les capacités descriptives des MG mais ont également suggéré des possibilités pour obtenir à peu de frais des grammaires TAG beaucoup plus compactes. En effet, il est bien connu que les grammaires TAG à large couverture ont tendance à exploser en nombre d'arbres, avec plusieurs milliers ou dizaines de milliers de (schémas d') arbres (Abeillé, 2002), ce qui rend très difficile l'analyse, même en exploitant des techniques de filtrage par les mots de la chaîne d'entrée. Les alternatives proposées passent par des techniques d'analyse des arbres pour retrouver et factoriser leurs parties communes (Carroll *et al.*, 1998) ou par la description de schémas de parcours multiples dans les arbres (Harbusch & Woch, 2004). Les méta-grammaires s'appuyant sur des descriptions factorisées nous permettent d'aller plus facilement dans la direction de tels arbres *factorisés*.

Le système DYALOG que nous utilisons pour la construction d'analyseurs syntaxiques peut gérer de tels arbres factorisés (Section 2). En conséquence, en parallèle avec la conception d'une méta-grammaire du français, nous avons étendu les possibilités descriptives des MG et les possibilités génératives de notre compilateur de MG (Section 3). Nous avons également complété notre environnement de travail pour les MG. La section 4 fournit quelques éléments d'information sur notre méta-grammaire et sur la grammaire résultante, en particulier au niveau de la compacité. Enfin, la section 5 fournit quelques résultats préliminaires pour notre analyseur.

2 Analyseurs hybrides TAG/TIG avec le système DYALOG

Le système DYALOG (Villemonte de la Clergerie, 2002) fournit un environnement de compilation et d'exécution d'analyseurs syntaxiques tabulaires (à la Earley) offrant la puissance d'un langage de programmation en logique. Il couvre divers formalismes syntaxiques, dont ceux utilisés dans notre expérience, à savoir les Grammaires d'Arbres Adjoints (TAG) et les Grammaires d'Insertion d'Arbres (TIG).

Les TAG (Joshi, 1987) sont formées d'arbres partiels d'analyse combinables par substitution et adjonction. Un nœud feuille étiqueté par un non-terminal peut être substitué par un arbre initial. Une adjonction insère le contenu d'un arbre auxiliaire β au niveau d'un nœud N, le sous-arbre

de racine N étant rattaché au niveau du *pied* f_β de β. Dans les FTAG, les nœuds sont décorés par une paire d'attributs `top` et `bot`, généralement exprimés comme des structures de traits.

Les TIG (Schabes & Waters, 1995) sont une variante des TAG restreignant les arbres auxiliaires de sorte qu'ils ne puissent s'insérer qu'à droite ou à gauche du nœud d'adjonction. Cette condition implique en particulier que les arbres auxiliaires aient leur *dorsale* (c.a.d. le chemin de la racine au pied) comme frontière gauche ou droite. L'intérêt majeur des TIG provient du fait qu'elles sont analysables, comme les CFG, avec une complexité en $O(n^3)$ alors que les TAG le sont en $O(n^6)$ où n dénote la longueur de la chaîne d'entrée. De plus, la plupart des grammaires TAG sont essentiellement TIG et il est en fait possible de construire des analyseurs syntaxiques hybrides TAG/TIG (Alonso & Díaz, 2003). DYALOG peut analyser une grammaire TAG pour identifier les parties TIG afin de construire de tels analyseurs hybrides TAG/TIG[1].

Pour les différents formalismes syntaxiques qu'il couvre, le système DYALOG permet, à l'intérieur des structures grammaticales, l'usage d'*opérateurs réguliers* tels que la disjonction, l'étoile de Kleene et l'entrelacement, ce dernier permettant d'indiquer un ordre libre entre des séquences de constituants (Nederhof *et al.*, 2003). Ces opérateurs ne changent pas le formalisme sous-jacent car ils peuvent en théorie être expansés et éliminés en introduisant de nouvelles structures grammaticales (arbres ou productions) et/ou de nouveaux non-terminaux. Néanmoins, le taux d'expansion peut être exponentiel en le nombre d'occurrences de ces opérateurs. Leur utilisation permet donc d'obtenir des grammaires beaucoup plus compactes et plus efficaces, car ces opérateurs sont utilisés sans expansion. D'autre part, il est à noter que l'usage de ces opérateurs rend plus naturel les forêts de dérivations en évitant l'usage de non-terminaux artificiels.

3 Étendre les MetaGrammaires

```
1 class collect_real_subject_canonical {
2     <: collect_real_subject;
3     $arg.extracted = value(~cleft);
4     S >> VSubj;  VSubj < V;   V >> postsubj;  VMod < postsubj;
5     node postsubj: [cat:N2,id:subject, type:subst, top:[wh:-,sat:+]];
6     - postsubj::agreement; postsubj = postsubj::N;
7     postsubj =>
8       node(Infl).bot.inv = value(+),
9       $arg.extracted = value(-), $arg.real = value(N2),
10      desc.extraction = value(~-),
11      node(V).top.mode=value(~infinitive|imperative|gerundive|participle);
12    ~ postsubj => node(Infl).bot.inv = value(~+);
13 }
```

Listing 1 – Exemple de classe

Le listing 1 illustre une classe fille `collect_real_subject_canonical` héritant de la classe parente `collect_real_subject`. Cette dernière décrit l'ensemble des réalisations possibles du sujet et est utilisée comme modèle pour les diverses réalisations du sujet en position canonique ou en extraction clivée[2]. La classe fille complète la classe parente pour le cas cano-

[1] Il est à noter que cette analyse ne garantit pas toujours l'équivalence entre analyseurs TAG et analyseurs hybrides TAG/TIG suite aux décorations et à des gestions différentes de l'adjonction, à savoir adjonction «chaînée» (sur les racines des arbres auxiliaires) pour les TAG contre adjonction multiple pour les TIG.

[2] Type « C'est de travailler qui me fatigue ! ».

nique, en précisant la position du sujet (sous `S` et devant le noyau verbal `V`) et en introduisant la notion de sujet post-verbal uniquement réalisable par un groupe nominal (`N2`).

Plus formellement, les méta-grammaires permettent une description syntaxique éclatée à l'aide de contraintes élémentaires regroupées en classes. Une classe peut hériter des contraintes de plusieurs classes parentes (**<:**, ligne 2) et peut également fournir une ressource ($+r$) ou requérir une ressource ($-r$, l. 6).

Les contraintes peuvent porter sur les nœuds (l. 4 et 6) incluant l'égalité **=**, la précédence **<** ainsi que les dominances immédiates **>>** et indirectes **>>+**. Les contraintes peuvent aussi porter sur les décorations des nœuds (l. 5) ou de la classe elle-même (**desc**, l. 10). Les décorations sont exprimées comme des structures de traits (l. 5) avec possibilité d'utiliser des disjonctions **|** et négations **~** sur des valeurs atomiques (l. 11) ainsi que des variables (**$arg**). Les contraintes sur les décorations s'expriment soit directement soit au travers d'équations entre chemins de traits ancrés sur des nœuds (l. 8), sur la classe elle-même (**desc**, ligne 10) ou sur des variables (l. 9). Des macros peuvent être utilisées pour nommer des valeurs ou des chemins. Enfin, il est possible de faire porter des contraintes sur le père d'un nœud N avec la notation « father (N)».

L'objectif du compilateur de méta-grammaire[3] est alors de croiser, par point fixe, les classes terminales (c.a.d. sans descendants) de manière à obtenir des classes *neutres* pour lesquelles chaque ressource fournie est consommée et réciproquement. Les contraintes sont accumulées lors des croisements et seules sont conservées les classes dont les contraintes accumulées, prenant en compte leurs conséquences logiques, sont satisfiables[4] . Les contraintes des classes neutres survivantes sont ensuite exploitées pour produire les structures grammaticales minimales, en l'occurrence des arbres pour les TAG.

Dans la formalisation standard des MG (Candito, 1999), une ressource peut être neutralisée au plus une fois pour produire une classe neutre. Cette restriction amène à dupliquer certaines classes pour nommer différemment la même ressource. Ainsi, pour exprimer qu'une classe décrivant les verbes a besoin de 2 arguments verbaux, il faut dupliquer une partie importante de la hiérarchie des classes pour deux ressources similaires `-varg1` et `-varg2`. Pour lever cette limitation, nous avons introduit la notion d'espace de noms et rompu la symétrie entre fournisseurs et consommateurs : une ressource peut maintenant être demandée dans un certain espace de nom ns (ns = `postsub` dans −postsubj**::**agreement, l. 6) et lors d'un croisement avec une classe fournisseuse $\mathcal{C}$ (ici, fournissant **+**agreement), les nœuds, variables et besoins de $\mathcal{C}$ sont alors plongés dans l'espace de nom ns (ici postsubj). Les espaces de noms permettent un usage beaucoup plus intensif du mécanisme de ressources et une bien meilleure factorisation des méta-grammaires. Les MG sont alors moins redondantes et plus faciles à maintenir.

Les décorations portées par les nœuds et la classe sont libres mais certaines ont néanmoins un statut spécial par rapport à la génération des arbres TAG. Pour les nœuds, on peut citer les traits `cat` pour la catégorie syntaxique, `type` pour le type de nœud, `lex` pour une valeur lexicale, `adj` pour indiquer le statut du nœud pour l'adjonction, `top` et `bot` comme arguments. Pour les classes, le trait `ht` indique l'hypertag qui sera associé aux arbres pour permettre l'ancrage avec les entrées lexicales (voir Section 4).

La possibilité d'engendrer des arbres *factorisés* résulte de divers mécanismes. En premier lieu, à côté des types standards de nœuds, il existe les types spéciaux `alternative` et `sequence`. Le trait `optional` permet de rendre optionnel un nœud tandis que le trait `star` permet de

[3]Développé sous le système DYALOG.

[4]Par exemple, le compilateur vérifie qu'un noeud ne précède pas son père.

rendre un nœud répétable, correspondant à une étoile de Kleene[5]. Enfin, lors de l'énumération des arbres minimaux vérifiant un ensemble de contraintes, le compilateur utilise l'opérateur d'*entrelacement* (##) pour rendre compte de sous-spécification de précédence entre nœuds frères. Ainsi, les contraintes «N **>>** N_1; N**>>** N_2; N **>>** N_3; N_1 **<** N_2» produisent le fragment d'arbre $N((N_1, N_2)\#\#N_3)$ indiquant que N_3 se positionne librement (avant, au milieu, après) par rapport à la séquence N_1, N_2. Pour favoriser l'obtention d'arbres TIG, le compilateur évite, dans la mesure du possible, d'utiliser l'opérateur d'entrelacement quand il couvre un nœud pied comme dans $R_\beta(N\#\#F_\beta)$. Dans ce cas, les différentes possibilités d'ordonnancement des nœuds sont examinées pour produire des arbres que l'on espère être TIG. Il est également possible d'assigner un rang à un nœud avec le trait `rank` et les valeurs `first` et `last`.

L'optionnalité fournie par l'emploi du trait `optional` n'est pas assez fine en pratique. L'emploi de *gardes* permet d'imposer des conditions à l'existence d'un nœud (l. 7) ou à sa non-existence (l. 12). Ces gardes s'expriment comme des expressions booléennes sur des équations entre chemins. Le compilateur de MG vérifie la satisfiabilité de ces gardes, éliminant les alternatives conduisant à des échecs et les équations devenues tautologiquement vraies. Les gardes restantes sont alors émises dans les arbres TAG pour être évaluées pendant l'analyse.

Outre les extensions des méta-grammaires et du compilateur, le travail de description a été facilité par le déploiement d'un environnement de travail adapté pour pouvoir aisément visualiser et tester. En premier lieu, nous disposons d'un mode Emacs pour les MG interagissant avec un outil graphique de visualisation de la hiérarchie des classes. Par ailleurs, la chaîne de traitement allant des méta-grammaires aux analyseurs produit des représentations intermédiaires sous formats XML[6] pouvant être visualisées, en particulier sous forme HTML pour les arbres, décorations et gardes. Les forêts de dérivations produites par notre analyseur sont également convertibles en XML et visualisables sous différentes formes, en particulier sous forme de dépendances. L'utilisation d'un serveur d'analyseurs[7] couplé à divers scripts facilite la conduite de tests sur corpus, pour mesurer divers paramètres (temps d'analyse, taux d'ambiguïté, taux de couverture, ...) et indiquer les différences entre 2 séries de tests. Enfin, il est possible de désactiver des classes[8] pour déboguer ou, à terme, pour obtenir des grammaires spécialisées. Ces diverses possibilités permettent un suivi fin des performances de la grammaire engendrée.

4 Anatomie de la grammaire produite

Grâce aux résultats décrits précédemment, nous avons pu rapidement concevoir une méta-grammaire du français engendrant une grammaire très compacte, comme le montrent les diverses tables de la figure 1. Ainsi, la grammaire ne comporte que 133 arbres, incluant 7 arbres construits manuellement. Elle est essentiellement TIG avec seulement 12 arbres auxiliaires enveloppants principalement utilisés pour gérer les diverses formes de guillemets[9]. La grammaire n'est pas totalement lexicalisée, avec un nombre assez important d'arbres sans ancre (mais

[5] À terme, la valeur du trait sera exploitée pour pouvoir spécifier un intervalle de répétition.

[6] Ces formats XML s'appuient de plus sur les propositions de normalisation, à savoir TAGML pour les TAG et FSR pour les structures de traits.

[7] Accessible en ligne sur `http://atoll.inria.fr/parserdemo`.

[8] Il est en fait possible d'activer ou désactiver de manière plus fine, en exprimant un ensemble de contraintes invalidant une classe.

[9] Pour être plus précis, ces arbres sont uniquement utilisés pour les guillemets autour de groupes, ceux autour de mots simples sont gérés avant analyse syntaxique. Le traitement proposé est clairement une source d'inefficacité pouvant peut-être être géré autrement. Par ailleurs, il est à noter que le compilateur MG a produit plus

possédant éventuellement des nœuds lexicaux), essentiellement utilisés pour des adjonctions[10]. Les arbres ancrés le sont surtout par les verbes mais ils ne représentent qu'une infime fraction d'un ensemble équivalent d'arbres TAG non factorisés. On voit que 7 arbres suffisent à couvrir un ensemble conséquent de constructions verbales « canoniques ». Ces résultats découlent d'un usage intensif de la factorisation dans les arbres, en particulier contrôlée par des gardes. L'étoile de Kleene est uniquement utilisée pour gérer la coordination tandis que les entrelacements proviennent essentiellement d'un ordre libre entre arguments du verbe (incluant le sujet post-verbal). Les arbres factorisés obtenus peuvent être relativement conséquents (jusqu'à 46 nœuds) mais la figure 1(e) montre néanmoins que la plupart des arbres restent simples.

Classes	Arbres	Init.	Aux.	Aux. Env.	Aux. Gauches	Aux. Droits
191	133=126+7	44	89	12	29	48

(a) Distribution par types d'arbres

non ancrés	v	coo	adv	adj	csu	prep	aux	np	nc	det	pro
50	27	12	10	8	4	3	2	2	1	1	1

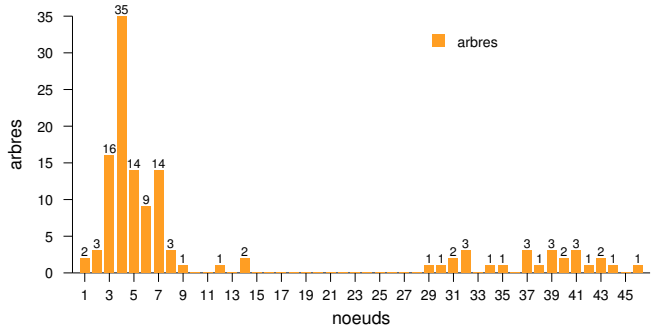
(b) Distribution par ancres

Canonique	Extr.	Actif	Passif	Quest.	Rel.	Clivées	Coord	Adv	Adj
7	19	19	6	4	4	11	12	14	11

(c) Distribution par phénomènes syntaxiques

Gardes	Disjonctions	Entrelacement	Étoiles de Kleene
820	92	26	13

(d) Distribution des factorisations



(e) Distribution des tailles d'arbres

FIG. 1 – Anatomie de la grammaire

La complexité des arbres factorisés est illustrée par la figure 2 représentant une vue simplifiée d'un des arbres verbaux canoniques pour la voix active. Cet arbre `#111` résulte du croisement de 25 classes terminales, comprend 43 nœuds plus 3 nœuds d'alternatives et 1 nœud d'entrelacement, et est contrôlé par 35 gardes[11]. Il est difficile d'obtenir le taux exact de factorisation atteint, mais voici néanmoins quelques paramètres indicatifs pour essayer de l'estimer :

d'arbres auxiliaires que nécessaire pour éviter d'avoir des pseudo-arbres enveloppants et que certains phénomènes syntaxiques pouvant produire des arbres enveloppants ont été bridés pour obtenir des arbres TIG.

[10]Cette non lexicalisation partielle est guidée par des raisons pragmatiques (limitation du nombre d'arbres) mais également linguistiques. Elle ne remet pas en cause la notion de domaine de localité sémantique des arbres TAG. Au contraire, l'accroche d'une participiale sur un nom, par exemple, est non lexicalisée car distincte (sémantiquement) de la construction d'une participiale.

[11]Un tel arbre avec toutes ses gardes et décorations serait extrêmement difficile à écrire à la main (2171 lignes de XML TAGML), ce qui justifie d'autant plus le recours à une méta-grammaire.

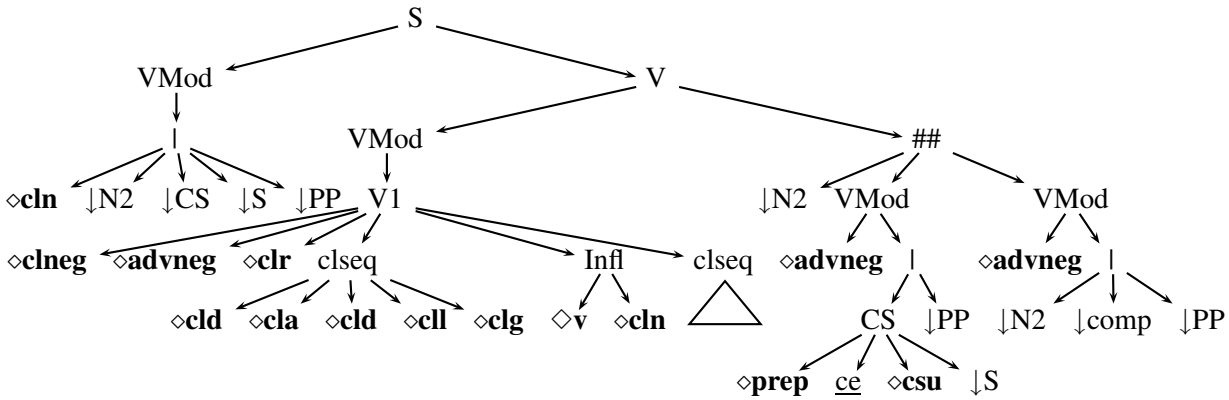


FIG. 2 – Arbre #111 (simplifié)

subjreal $= 5$ choix de réalisations pour un sujet pré-verbal : `cln` (clitique nominatif), `N2` (groupe nominal), `CS` (phrase), `S` (infinitive), `PP` (infinitive introduite par « d' ») ;

subjpos choix entre « pas de sujet », « sujet pré-verbal », « sujet pré-verbal et clitique post-verbal », « clitique post-verbal », « sujet post-verbal » ;

argspos $= 3$ entrelacements entre 2 arguments post-verbaux et un sujet post-verbal, tous optionnels ;

clseqpos $= X$ choix entre 2 séquences de clitiques pré ou post verbaux optionnels : `cld`, `cla`, `cld`, `cll`, `clg` ;

clopt $= 2$ clitiques pré-verbaux optionnels : `clneg` (« ne »), `clr` (réflexif) ;

neg $= 3$ positions de négation (advneg « que » sur les arguments verbaux ou advneg pré-verbal pour les infinitives) ;

arg1real $= 3$ réalisations possibles pour arg1 : `N2` (objet), `comp` (attribut), `PP` (groupe prépositionnel nominal, adjectival, infinitif, complétif, ...) ;

arg2real $= 2$ réalisations possibles (au moins) pour arg2 (CS,PP) ;

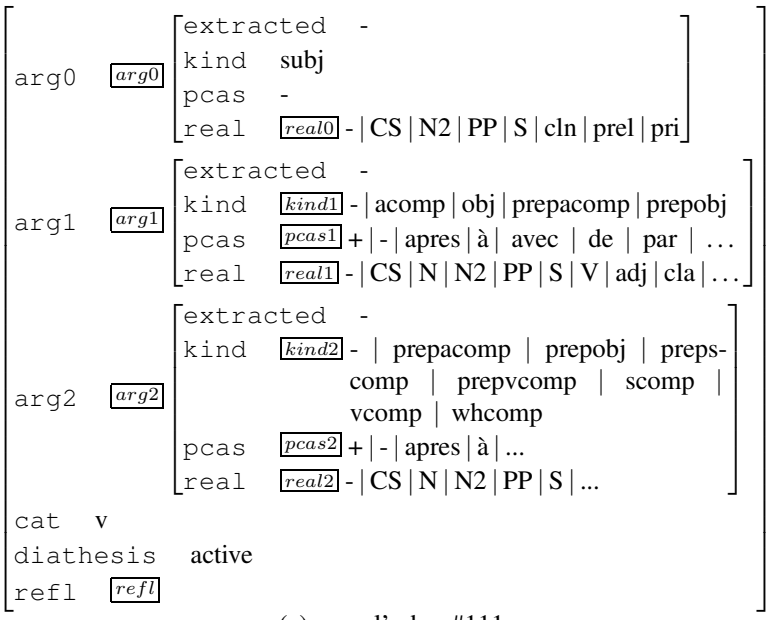
Divers calculs approchés donnent un taux de factorisation entre 10 000 et 100 000, sans même prendre complètement en compte les disjonctions apportées par certaines gardes. Ce taux d'expansion astronomique est bien entendu très largement au-dessus de la réalité si on prend mieux en compte la satisfiabilité des gardes. Néanmoins, même un taux d'expansion plus réaliste de l'ordre de 100 ou 1000 montre l'intérêt des ces structures factorisées, difficilement constructibles manuellement sans le support apporté par les méta-grammaires. De plus, les expériences menées ont montré que l'analyse reste efficace même avec des structures aussi complexes[12].

Un arbre comme l'arbre `#111` couvre de fait de très nombreuses constructions syntaxiques, bien plus en général que celles autorisées pour un mot donné. L'ancrage entre mots et arbres, ainsi que la sélection des constructions autorisées pour un mot se fait par unification entre des structures de traits appelées *hypertag* associés aux arbres et aux mots (Kinyon, 2000)[13]. Ainsi la figure 3(a) donne l'hypertag produit pour l'arbre `#111` tandis que la figure 3(b) montre un des hypertags mentionnés pour « promettre » dans notre lexique. Parmi toutes les constructions possibles pour l'arbre `#111`, l'unification sélectionne uniquement les constructions verbales canoniques autorisant un objet optionnel et un groupe prépositionnel optionnel introduit par

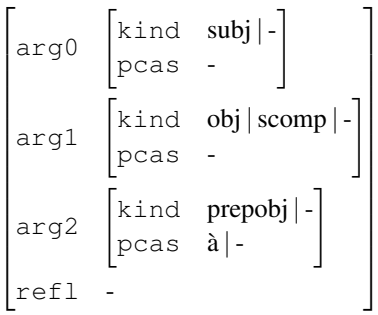
[12]En général, les mauvaises performances résultent plutôt d'arbres sous-contraints faisant exploser le nombre d'analyses.

[13]Ce mécanisme d'ancrage par hypertags rend caduque la notion trop rigide de famille d'arbres présente dans l'architecture XTAG (Doran *et al.*, 1994).

« à »[14]. Le lien entre hypertag $\mathcal{H}$ et des constructions syntaxiques se fait grâce aux variables présentes dans $\mathcal{H}$ et dans les décorations des nœuds ou dans les équations des gardes.



(a) pour l'arbre #111



(b) pour «promettre»

FIG. 3 – Hypertags

5 Expériences

L'analyseur hybride TAG/TIG compilé à partir de la grammaire a été testé sur divers corpus tout au long de la phase de développement et pendant la campagne EASY. Les analyses s'effectuent sur des treillis de mots (pour gérer les ambiguïtés morpho-syntaxiques et les mots inconnus) en s'appuyant sur un lexique de plus de 400 000 formes fléchies fournissant des informations de sous-catégorisation pour les verbes. Nous n'avons pas utilisé d'étiqueteur morpho-syntaxique. L'analyseur s'appuie sur une stratégie d'analyse tabulaire descendante gauche-droite et peut rendre soit une analyse complète de la phrase soit un ensemble d'analyses partielles couvrant au mieux l'entrée. Les analyses sont extraites sous forme de forêts partagées de dérivations, convertibles en forêts partagées de dépendances. Ces forêts nous servent de base pour calculer un *taux moyen d'ambiguïté par mot* α défini comme le nombre moyen d'arcs de dépendances atteignant un mot moins un[15].

Corpus	**#phrases**	**% couv.**	**temps moyen (s)**	**temps médian (s)**	**ambiguïté**
EUROTRA / OLD	334	95.80 / 89.22	1.81 / 0.70	1.27 / 0.54	0.7 / 0.3
TSNLP / OLD	1661	93.38 / 86.15	0.72 / 0.43	0.56 / 0.33	0.4 / 0.2
MD10x20 / OLD	5000	63.18 / 43.06	2.85 / 1.97	1.80 / 1.30	0.8 / 0.5
EASY	34438	42.45 / -	5.55 / -	1.61 / -	0.6 / -

TAB. 1 – Résultats (avec un *timeout* de 100s)

[14] La sélection des constructions avec une complétive (`scomp`) se fait avec un autre arbre.

[15] Pour une analyse non-ambiguë, tout mot sauf la « tête » de la phrase est atteignable par une seule dépendance. Le nombre maximal d'analyses pour un taux d'ambiguïté α et une phrase de longueur n est en $O((1+\alpha)^n)$.

La table 1 fournit des résultats d'analyses complètes de 2 versions successives de l'analyseur pour les jeux de tests EUROTRA et TSNLP ainsi que pour MD10x20, un corpus journalistique de phrases de longueur comprise entre 10 et 20 extraites (naïvement) du « *Monde Diplomatique* » et pour le corpus fourni pour la campagne EASY (couvrant divers styles : journalistique, littéraire, oral, mail, médical, questions/réponses). Les résultats de couverture sont excellents sur les jeux de tests, en particulier à cause d'un vocabulaire relativement restreint pour lequel notre lexique est complet. Sur le corpus MD10x20 qui est relativement homogène et pour lequel un minimum d'adaptation du lexique a été effectué, les résultats restent honorables. Les résultats sont moins bons pour EASY qui est très hétérogène[16]. Ce manque de couverture traduit bien évidemment des manques dans la méta-grammaire, en particulier sur les coordinations complexes, les superlatives et les comparatives, ainsi que sur les cadres de sous-catégorisation pour les catégories non-verbales et sur certaines articulations de phrase. Cependant, le manque de couverture provient également de notre lexique qui est très récent et ne fournit pas nécessairement des informations syntaxiques complètes voire correctes pour tous les mots[17].

La table 1 fournit aussi des résultats pour une version antérieure de l'analyseur (OLD) qui illustrent l'importance du suivi constant des grammaires. En effet, nous avons effectué, sans réel contrôle, des modifications de dernière minute avant EASY pour essayer d'améliorer la couverture (extension des clivées, généralisation abusive des incises, articulation des phrases par la ponctuation, gestion naïve des verbes support, . . .). Ces modifications ont bien augmenté la couverture, mais, mal contrôlées, elles ont fait doubler les taux d'ambiguïté et les temps d'analyse (avec en première approximation, une relation linéaire entre temps et taux d'ambiguïté).

Enfin, sans corpus de référence, il nous est impossible pour l'instant de fournir des résultats concernant la précision des analyses (complètes ou partielles). Nous avons effectué de nombreuses vérifications manuelles sur les vues graphiques des forêts mais attendons maintenant les résultats de la campagne EASY pour avancer.

6 Conclusion

Notre méta-grammaire est encore loin d'être complète mais l'expérience montre néanmoins que les méta-grammaires rendent possible le développement rapide de grammaires à relativement large couverture. Il est à noter que ce développement a été en partie freiné par le manque d'information dans le lexique, en particulier pour avoir une discrimination plus fine des adverbes et pour traiter les sous-catégorisations des adjectifs et des noms.

Les extensions apportées aux méta-grammaires ainsi que les améliorations de notre environnement de travail se sont révélées très utiles. Néanmoins, concevoir une méta-grammaire reste un exercice délicat demandant une solide expertise linguistique et une utilisation systématique d'outils de tests et de visualisation. Il nous semble aussi souhaitable d'ajouter de nouveaux types de contraintes, même si elles peuvent s'exprimer à l'aide des contraintes actuelles, comme des contraintes d'exclusion entre nœuds, des contraintes de cardinalité pour exprimer des règles topologiques, ou des contraintes de rangs exprimables dans les gardes. Pour aller dans le sens de grammaires paramétrables (autorisant divers niveaux de langue) ou pour aller vers des formalismes cibles distincts, il serait utile de regrouper les contraintes par *contextes* à l'intérieur des

[16] On peut aussi préciser que les phrases pour EASY sont en moyenne plus longues.

[17] Mais nous exploitons progressivement les résultats d'analyse pour repérer et corriger les entrées incorrectes ou incomplètes.

classes de manière à pouvoir plus facilement n'en exploiter qu'une partie lors de la compilation (en sélectionnant un ensemble de contextes).

La factorisation des arbres, rendue possible par l'emploi de gardes et d'opérateurs réguliers, nous semble une approche générique extrêmement prometteuse pour contrôler l'explosion combinatoire du nombre de structures grammaticales produites, permettant ainsi de construire des analyseurs syntaxiques plus efficaces. Les arbres factorisés peuvent être complexes mais leur description au niveau de la méta-grammaire reste simple.

Le formalisme cible TAG que nous avons utilisé est judicieux mais néanmoins pas suffisamment puissant pour exprimer élégamment certains phénomènes syntaxiques comme les incises ou certaines extractions (comme l'extraction de génitifs dans « *de qui lis-tu un livre* »). Nous envisageons d'évoluer vers des formalismes cibles permettant d'exprimer plus de sous-spécification dans les arbres, comme par exemple les *Local Multi Component TAG*, avec l'ambition, à terme, de réduire la distance entre les méta-grammaires et le formalisme cible.

Les outils mentionnés dans cet article ainsi que la méta-grammaire sont librement disponibles[18].

Références

ABEILLÉ A. (2002). *Une grammaire électronique du français*. Paris : CNRS Editions.

ALONSO M. A. & DÍAZ V. J. (2003). Variants of mixed parsing of TAG and TIG. *Traitement Automatique des Langues (T.A.L.)*, **44**(3), 41–65.

CANDITO M.-H. (1999). *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées*. PhD thesis, Université Paris 7.

CARROLL J., NICOLOV N., SMETS M., SHAUMYAN O. & WEIR D. (1998). Grammar compaction and computation sharing in automata-based parsing. In *Proceedings of Tabulation in Parsing and Deduction (TAPD'98)*, p. 16–25, Paris (FRANCE).

CLÉMENT L. & KINYON A. (2003). Generating parallel multilingual LFG-TAG grammars from a metaGrammar. In *Proc. of ACL'03*.

DORAN C., EGEDI D., HOCKEY B. A., SRINIVAS B. & ZAIDEL M. (1994). XTAG system — a wide coverage grammar for English. In *Proc. of the 15th International Conference on Computational Linguistics (COLING'94)*, p. 922–928, Kyoto, Japan.

GAIFFE B., CRABBÉ B. & ROUSSANALY A. (2003). Représentation et gestion du lexique d'une grammaire d'arbres adjoints. *Traitement Automatique des Langues (T.A.L.)*, **44**(3).

HARBUSCH K. & WOCH J. (2004). Integrated natural language generation with schema-tree adjoining grammars. In C. HABEL & E. THOMAS PECHMANN, Eds., *Language Production*. Mouton De Gruyter.

JOSHI A. K. (1987). An introduction to tree adjoining grammars. In A. MANASTER-RAMER, Ed., *Mathematics of Language*, p. 87–115. Amsterdam/Philadelphia : John Benjamins Publishing Co.

KINYON A. (2000). Hypertags. In *Proc. of COLING*, p. 446–452.

NEDERHOF M.-J., SATTA G. & SHIEBER S. (2003). Partially ordered multiset context-free grammars and free-word-order parsing. In *In 8th International Workshop on Parsing Technologies (IWPT'03)*, p. 171–182.

SCHABES Y. & WATERS R. C. (1995). Tree insertion grammar : a cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Fuzzy Sets Syst.*, **76**(3), 309–317.

VILLEMONTE DE LA CLERGERIE E. (2002). Construire des analyseurs avec DyALog. In *Proc. of TALN'02*.

[18] Sur `http://atoll.inria.fr/packages/packages.html`.