Natural Language Generation for Language Learning

Laura Haide Perez

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Génération automatique de phrases pour l'apprentissage des langues

(Natural Language Generation for Language Learning)

THÈSE

présentée et soutenue publiquement le 19 avril 2013

pour l'obtention du

Doctorat de l'Université de Lorraine

(Mention Informatique)

par

Laura Haide PEREZ

Composition du jury

Rapporteurs :	Karin HARBUSCH	Professeur, Universität Koblenz-Landau, Allemagne
	Richard POWER	Professeur, Open University, Milton Keynes, R.U.
Examinateurs :	Claire GARDENT	Directeur de Recherches CNRS, LORIA Nancy, France
	Guy PERRIER	Professeur, Université de Lorraine, Nancy, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Acknowledgements

I would like to use this space to thank many people who have contributed to this thesis work in one way or another.

First of all, I want to thank my supervisor Claire for giving me the opportunity to work in this topic, for her constant support, for giving me space but at the same time being always ready to give advice, and for correcting many papers, slides and this manuscript as well. Thank you Claire for all I have learnt in the past three years.

I am indebted to the Université de Lorraine for the financial support and the doctoral school IAEM as well as LORIA for the necessary orchestration altogether making my doctoral studies possible.

Many thanks also go to the members of the jury, Karin Harbusch, Guy Perrier and Richard Power for their constructive comments and interesting feedback and discussions about the research work in this thesis.

A special thank you to my Allegro colleagues Alex, Céline, German, Ingrid and Marilisa for discussions about work and their collaboration as well as Elise and Natalia for their help with the evaluations and Samuel and Nadia for facilitating the use of IFLEG with French language students. Thanks also to Céline for reviewing the French parts, Lina for agreeing to read some chapters and Shashi for the triangle tree drawings. Many thanks to all the members of the Synalp/Talaris group for the comfortable work environment. Thank you very much to Kristina Striegnitz for kindly facilitating me the sources for the drawing of the dependency tree.

I would like to thank my friends and extended family, here and in Argentina, for sharing with me recreation time, mails, visits, home-made scarves and many good things and for enthusiastically encouraging me during these years.

I am infinitely grateful to my family for their unconditional support. I thank my parents, Nilda and Ricardo, for being an unbeatable example. They are strong, loving and intelligent; I feel very proud of them. Thanks to my brother Mariano for his care and for pointing me out to those simple big things about life. I want to thank Yann for staying by my side; his love and patience make me happy.

Thank you to those I have forgotten to mention.

I dedicate this thesis to Nilda, Ricardo, Mariano, Tata and Yann.

Laura

Résumé

Mots-clés: Grammaire d'Arbres Adjoints à Structures de Traits (FB-TAG), Réalisateur de Surface (RS), Optimisation de la Réalisation de Surface, Grammaire d'Arbres Réguliers à Structure de Traits (FB-RTG), Représentations sémantiques plates et sous-spéciées, Génération Automatique de Langue Naturelle (GLN), Apprentissage Assisté par Ordinateur (CALL), Création (Semi-)automatique d'exercices de grammaire.

Dans ces travaux, nous explorons comment les techniques de Générations Automatiques de Langue Naturelle (GLN) peuvent être utilisées pour aborder la tâche de génération (semi-)automatique de matériel et d'activités dans le contexte de l'apprentissage de langues assisté par ordinateur. En particulier, nous montrons comment un Réalisateur de Surface (RS) basé sur une grammaire peut être exploité pour la création automatique d'exercices de grammaire. Notre réalisateur de surface utilise une grammaire réversible étendue, à savoir *SemTAG*, qui est une Grammaire d'Arbre Adjoints à Structure de Traits (FB-TAG) couplée avec une sémantique compositionnelle basée sur l'unification. Plus précisément, la grammaire FB-TAG intègre une représentation plate et sous-spécifiée des formules de Logique de Premier Ordre (FOL).

Dans la première partie de la thèse, nous étudions la tâche de réalisation de surface à partir de formules sémantiques plates et nous proposons un algorithme de réalisation de surface basé sur la grammaire FB-TAG optimisé, qui supporte la génération de phrases longues étant donné une grammaire et un lexique à large couverture. L'approche suivie pour l'optimisation de la réalisation de surface basée sur FB-TAG à partir de sémantiques plates repose sur le fait qu'une grammaire FB-TAG peut être traduite en une Grammaire d'Arbres Réguliers à Structure de Traits (FB-RTG) décrivant ses arbres de dérivation. Le langage d'arbres de dérivation de la grammaire TAG constitue un langage plus simple que le langage d'arbres dérivés, c'est pourquoi des approches de génération basées sur les arbres de dérivation ont déjà été proposées. Notre approche se distingue des précédentes par le fait que notre encodage FB-RTG prend en compte les structures de traits présentes dans la grammaire FB-TAG originelle, ayant de ce fait des conséquences importantes par rapport à la sur-génération et la préservation de l'interface syntaxe-sémantique. L'algorithme de génération d'arbres de dérivation que nous proposons est un algorithme de type Earley intégrant un ensemble de techniques d'optimisation bien connues: tabulation, partage-compression (sharing-packing) et indexation basée sur la sémantique.

Dans la seconde partie de la thèse, nous explorons comment notre réalisateur de surface basé sur *SemTAG* peut être utilisé pour la génération (semi-)automatique d'exercices de grammaire. Habituellement, les enseignants éditent manuellement les exercices et leurs solutions et les classent au regard de leur degré de difficulté ou du niveau attendu de l'apprenant. Un courant de recherche dans le Traitement Automatique des Langues (TAL)

pour l'apprentissage des langues assisté par ordinateur traite de la génération (semi-)automatique d'exercices. Principalement, ces travaux s'appuient sur des textes extraits du Web, utilisent des techniques d'apprentissage automatique et des techniques d'analyse de textes (par exemple, analyse de phrases, POS tagging, etc.). Ces approches confrontent l'apprenant à des phrases qui ont des syntaxes potentiellement complexes et du vocabulaire varié. En revanche, l'approche que nous proposons dans cette thèse aborde la génération (semi-)automatique d'exercices du type rencontré dans les manuels pour l'apprentissage des langues. Il s'agit, en d'autres termes, d'exercices dont la syntaxe et le vocabulaire sont faits sur mesure pour des objectifs pédagogiques et des sujets donnés. Les approches de génération basées sur des grammaires associent les phrases du langage naturel avec une représentation linguistique fine de leur propriété morpho-syntaxiques et de leur sémantique grâce à quoi il est possible de définir un langage de contraintes syntaxiques et morpho-syntaxiques permettant la sélection de phrases souches en accord avec un objectif pédagogique donné. Cette représentation permet en outre d'opérer un post-traitement des phrases sélectionées pour construire des exercices de grammaire. Nous montrons comment les exercices de grammaire de type à trous, de reconstitution ou de reformulation de phrases peuvent être automatiquement produits. L'approche a été intégrée dans le jeux sérieux I-FLEG (Interactive French Learning Game, Jeu interactif pour l'apprentissage du français) et a été évaluée à la fois par l'analyse des interactions avec des joueurs en ligne et en collaboration avec des enseignants.

Abstract

Keywords: Feature-Based Tree Adjoining Grammars (FB-TAG), Surface Realisation (SR), Surface Realisation Optimisation, Featured-Based Regular Tree Grammar (FB-RTG), flat and underspecified semantic representations, Natural Language Generation (NLG), Computer-Assisted Language Learning (CALL), (Semi-)automatic authoring of grammar exercises.

In this work, we explore how Natural Language Generation (NLG) techniques can be used to address the task of (semi-)automatically generating language learning material and activities in Camputer-Assisted Language Learning (CALL). In particular, we show how a grammar-based Surface Realiser (SR) can be usefully exploited for the automatic creation of grammar exercises. Our surface realiser uses a wide-coverage reversible grammar namely *SemTAG*, which is a Feature-Based Tree Adjoining Grammar (FB-TAG) equipped with a unification-based compositional semantics. More precisely, the FB-TAG grammar integrates a flat and underspecified representation of First Order Logic (FOL) formulae.

In the first part of the thesis, we study the task of surface realisation from flat semantic formulae and we propose an optimised FB-TAG-based realisation algorithm that supports the generation of longer sentences given a large scale grammar and lexicon. The approach followed to optimise TAG-based surface realisation from flat semantics draws on the fact that an FB-TAG can be translated into a Feature-Based Regular Tree Grammar (FB-RTG) describing its derivation trees. The derivation tree language of TAG constitutes

a simpler language than the derived tree language, and thus, generation approaches based on derivation trees have been already proposed. Our approach departs from previous ones in that our FB-RTG encoding accounts for feature structures present in the original FB-TAG having thus important consequences regarding over-generation and preservation of the syntax-semantics interface. The concrete derivation tree generation algorithm that we propose is an Earley-style algorithm integrating a set of well-known optimisation techniques: tabulation, sharing-packing, and semantic-based indexing.

In the second part of the thesis, we explore how our *SemTAG*-based surface realiser can be put to work for the (semi-) automatic generation of grammar exercises. Usually, teachers manually edit exercises and their solutions, and classify them according to the degree of dificulty or expected learner level. A strand of research in (Natural Language Processing (NLP) for CALL addresses the (semi-)automatic generation of exercises. Mostly, this work draws on texts extracted from the Web, use machine learning and text analysis techniques (e.g. parsing, POS tagging, etc.). These approaches expose the learner to sentences that have a potentially complex syntax and diverse vocabulary. In contrast, the approach we propose in this thesis addresses the (semi-) automatic generation of grammar exercises of the type found in grammar textbooks. In other words, it deals with the generation of exercises whose syntax and vocabulary are tailored to specific pedagogical goals and topics. Because the grammar-based generation approach associates natural language sentences with a rich linguistic description, it permits defining a syntactic and morpho-syntactic constraints specification language for the selection of stem sentences in compliance with a given pedagogical goal. Further, it allows for the post processing of the generated stem sentences to build grammar exercise items. We show how Fill-in-the-blank, Shuffle and Reformulation grammar exercises can be automatically produced. The approach has been integrated in the Interactive French Learning Game (I-FLEG) serious game for learning French and has been evaluated both based in the interactions with online players and in collaboration with a language teacher.

Contents

Génération automatique de phrases pour l'apprentissage des langues

This chapter presents a summary of the thesis, in French.
Ce chapitre présente un résumé en français de la thèse.

Sommaire

Cette thèse aborde l'utilisation des techniques de génération automatique de texte (NLG, Natural Language Generation) pour l'apprentissage des langues assisté par ordinateur (CALL, Computer-Assisted Language Learning). Nous montrons, en particulier, comment un réalisateur de surface (SR, Surface Realiser) basé sur une grammaire d'arbres adjoints peut être utilisé afin d'automatiser la génération d'exercices de grammaire pour l'apprentissage des langues. Le réalisateur de surface utilise une grammaire réversible à large couverture dénommée *SemTAG*, une grammaire d'arbres adjoints à structures de traits (FB-TAG, Feature-Based Tree Adjoining Grammar) couplée avec une sémantique compositionnelle basée sur l'unification.

La présente thèse se décompose en deux parties:

- Dans la première partie, nous examinons la tâche de génération de phrases à partir de formules sémantiques et proposons un algorithme optimisé qui permet de générer des phrases longues à partir d'une grammaire et d'un lexique à large couverture.

- Dans la seconde partie, nous étudions comment notre réalisateur de surface

basé sur *SemTAG* peut être utilisé pour la génération d'exercices de grammaire dont la syntaxe et le vocabulaire peuvent être contrôlés. Nous proposons une approche qui s'appuie sur les caractéristiques spécifiques aux structures linguistiques produites par le réalisateur de surface. D'une part, la grammaire constitue une ressource linguistique riche et précise décrivant les expressions de la langue naturelle. Cela permet la génération de phrases qui satisfont à certaines contraintes syntactiques et morpho-syntactiques comme par exemple, les contraintes imposées par un but pédagogique comme *l'apprentissage de la voix passive*. En outre, les riches informations linguistiques associées au texte généré par notre réalisateur de surface permettent un traitement fin permettant de créer des items d'exercices de types textes à trous et de reconstitution ou de reformulation de phrases. D'autre part, les entrées sous-spécifiées et par conséquent les différentes phrases produites par notre réalisateur de surface permettent la production automatique, à partir de peu d'entrées, d'une variété d'exercices syntaxiquement et morpho-syntaxiquement variés.

L'objectif de la génération automatique de texte en langage naturel est de produire du texte compréhensible en langage humain à partir de données. Ce procédé est guidé par un but communicatif, basé sur une source d'information (les données), et comporte une série d'étapes ou de sous-tâches. Traditionnellement, ces sous-tâches sont conçues et organisées dans une séquence ou "pipeline", permettant de gérer des décisions stratégiques "quoi dire" et des décisions tactiques "comment le dire". Une fois que le contenu ou sens à exprimer en langage naturel a été déterminé, le composant tactique effectuera différents choix comme par exemple les mots et les constructions syntactiques à utiliser pour exprimer ce contenu en langage naturel.

En particulier, le module du réalisateur de surface d'un système de génération, généralement le dernier module de ce pipeline, transforme une spécification linguistique abstraite en une expression en langue naturelle. C'est à dire qu'il connaît le langage ciblé comme par exemple l'ordre des mots. Il existe différents niveaux d'abstraction dans la spécification de l'entrée du réalisateur. Par exemple, l'entrée peut être un arbre de dépendance où les rôles syntactiques ainsi que les mots-outils ont été spécifiés. En fonction du degré de spécification de son entrée, le réalisateur de surface peut procéder de façon (presque) déterministe ou au contraire, produire plusieurs réponses en prenant des décisions variables par rapport à la manière d'exprimer la représentation sémantique. Dans cette thèse, nous supposons une entrée sémantique (p.ex. une forme logique) et plus précisément, des formules de sémantique à récursion minimale (MRS, Minimal Recursion Semantics). Étant donné la MRS décrite en (1a), la tâche du réalisateur de surface est de produire des phrases

telles que (1b-c).

(1) a. $\{l_0 : named(t,\ Tex),\ l_0 : indiv(t,\ m,\ sg),\ qeq(TR,\ l_0),\ l_1 : properq(t,\ TR,\ TS),$
$l_2 : le(u,\ CR,\ CS),\ qeq(CR,\ l_3),\ l_3 : universite(u),\ l_3 : indiv(u,\ f,\ sg),$
$l_4 : travailler(e,\ t,\ u),\ l_4 : event(e,\ pres,\ indet,\ ind)\}$

b. *Tex travaille à l'université.* (Tex works at the university)

c. *C'est Tex qui travaille à l'université.* (It is Tex who works at the university)

La réalisation de surface à partir d'une formule sémantique plate (p.ex. un sac de prédicats comme illustré dans l'exemple (1a)) est une tâche de complexité exponentielle. Brew (1992) et Koller and Striegnitz (2002) fournissent des preuves formelles indiquant qu'elle appartient à la classe des problèmes NP-complets. Plusieurs techniques d'optimisation ont été proposées pour améliorer les temps d'exécution dans la pratique. Notre objectif est l'optimisation du réalisateur de surface basé sur la grammaire FB-TAG. A cette fin, nous suivons l'idée de Koller and Striegnitz (2002) consistant à utiliser les arbres de dérivation de la grammaire TAG pour la génération. Cependant, en nous appuyant sur une traduction bien définie de FB-TAG vers une grammaire d'arbres réguliers basés sur les traits (FB-RTG ou Feature-Based Regular Tree Grammar, [Schmitz and Le Roux, 2008]) pour décrire le langage d'arbre de dérivation de la grammaire FB-TAG, nous différons de cette approche.

Cette traduction conserve toute l'information sémantique, syntactique et morphosyntactique de la grammaire originelle ayant, de ce fait, d'importantes conséquences qui distinguent notre approche des précédentes. En préservant toutes les informations linguistiques, l'encodage FB-RTG préserve l'interface syntaxe/sémantique et fournit une grammaire exacte des arbres de dérivation FB-TAG. Nous développons un algorithme de réalisation de surface basé sur la grammaire FB-RTG qui intègre plusieurs techniques pour optimiser la réalisation de surface.

Traditionnellement, la génération de textes a été utilisée entre autres pour (i) générer des rapports (par exemple pour générer des textes à partir de bases de données contenant des données issues d'appareil de mesures), (ii) pour générer des descriptions à partir d'une base de connaissance et (iii) pour exprimer en langue naturelle la sortie d'un gestionnaire de dialogue (dialogue manager). Les domaines d'application sont aussi variés: domaine médical, prévisions météorologiques, manuels d'instructions, verbalisation d'instructions à l'intérieur de mondes virtuels, entre autres. De plus, d'autres types d'applications de traitement automatique de langue (NLP, Natural Language Processing) tels que le résumé automatique de texte, la simplification de texte et la génération automatique de questions peuvent aussi impliquer une étape finale de re-génération. Dans une moindre mesure, les techniques

de génération automatique de textes ont également été utilisées dans le contexte de l'enseignement intelligent des langues assisté par ordinateur (ICALL, Intelligent Computer-Assisted Language Learning).

Une grande variété de travaux dans le domaine du traitement automatique de langue (NLP) et d'ICALL ont été effectués au cours des dernières années. Les techniques NLP ont principalement été utilisées pour contribuer à la création de contenus et d'activités d'apprentissage ou pour évaluer les résultats de l'apprenant et générer un retour approprié. Par exemple, ALICE-chan ([Levin and Evans, 1995]) est un tuteur intelligent de langage pour l'apprentissage du Japonais qui utilise la grammaire lexicale-fonctionnelle (LFG, Lexical Functional Grammar) pour l'analyse des phrases. Il permet ainsi d'assister les instructeurs dans la création d'exercices et offre la possibilité d'évaluer les réponses des apprenants à ces exercices. ALICE-chan propose une interface pour la création d'exercices où les instructeurs entrent un texte correspondant au contexte, aux questions et aux réponses de ces exercices. La réponse est analysée par le module NLP générant, à partir de celle-ci, une structure de traits synthétisant des traits syntactiques et morpho-syntactiques qui seront utilisés plus tard pour évaluer les réponses de l'apprenant qui seront analysées d'une façon similaire. Il existe d'autres systèmes de tutorat comme TAGARELA ([Amaral and Meurers, 2011]). Celui-ci inclut des activités similaires à celles issues de livres d'apprentissage de langues: lecture et compréhension orale, description d'images, reformulations, textes à trous et exercices de vocabulaire. Différents outils de type NLP (p.ex. segmenteur ou analyseur de phrases) sont déployés dans son architecture. Celle-ci est centrée sur le traitement des réponses de l'apprenant et sur la production de retours appropriés à partir des modèles experts (connaissance du langage naturel), des modèles d'activités et des modèles d'apprenants.

Parmi les différentes applications ICALL, il y a des systèmes qui constituent des aides à l'écriture tels qu'ICICLE ([Michaud *et al.*, 2000]) ou des assistants pour la lecture tels que CALLE ([Rypa and Feuerman, 1995]) qui utilisent des techniques d'analyse de phrases. ICICLE utilise des techniques d'analyse de phrases pour analyser les réponses de l'apprenant, tandis que CALLE utilise des techniques d'analyse de phrases pour analyser des documents sélectionnés par l'apprenant et ainsi fournir des informations relatives aux constructions linguistiques présentes dans ces documents. Chacune à leur manière, ces deux applications ont pour but de mettre l'accent sur la connaissance et l'apprentissage des constructions grammaticales de la langue ciblée. WERTI ([Meurers *et al.*, 2010]) et VISL ([Bick, 2005]) sont deux autres applications ont dont le but principal est de promouvoir la sensibilisation linguistique. WERTI est une application d'amélioration de texte, utilisant des outils NLP pour re-

connaître et mettre en valeur les différentes caractéristiques grammaticales dans des documents sélectionnés sur le Web. VISL est un outil doté d'une interface graphique interactive pour l'apprentissage de syntaxe, qui utilise des outils NLP pour l'analyse.

Certains travaux se sont concentrés sur l'édition automatique d'exercices pour l'apprentissage des langues ([Mitkov *et al.*, 2006; Heilman and Eskenazi, 2007; Karamanis *et al.*, 2006; Chao-Lin *et al.*, 2005; Coniam, 1997; Sumita *et al.*, 2005; Simon *et al.*, 2010; Lin *et al.*, 2007; Lee and Seneff, 2007]). Plus particulièrement, certaines propositions ont pour objet la proposition d'exercices de grammaire (p.ex. [Aldabe *et al.*, 2006; Chen *et al.*, 2006]). En général, ces approches reposent sur des techniques d'apprentissage automatique et génèrent des activités pour l'apprentissage avancé.

Dans CALL, il existe des outils d'édition tels que *Hot Potatoes*[1] ([Winke and MacGregor, 2001]) qui n'utilisent pas des techniques NLP. Ils sont également appelés outils d'édition basés sur des modèles parce qu'ils fournissent un ensemble d'activités types que le professeur de langues peut utiliser pour créer des exercices. Cependant, le contenu pour chaque exercice, c'est-à-dire le texte de l'exercice, la ou les réponse(s) attendue(s) et le retour utilisateur doivent être manuellement par le professeur de langues.

En résumé, il existe des applications CALL ou les exercices pour l'apprentissage sont édités à la main, ou des applications ICALL dans lesquelles la plupart des techniques de TAL dédiées à la création (semi-)automatique de matériel pour l'apprentissage sont basée sur l'analyse de texte. Dans le dernier cas, le contenu textuel utilisé pour créer des activités d'apprentissage est soit fourni par le professeur de langues, soit collecté automatiquement à partir du Web. Dans cette thèse, nous montrons que la génération automatique de texte est une approche rassemblant les caractéristiques appropriées pour la génération (semi-)automatique d'exercices de grammaire pour l'apprentissage des langues.

Nous exploitons la grammaire paraphrastique à large couverture FB-TAG qui fournit une riche description linguistique du langage naturel en associant des expressions du langage naturel avec des syntaxes et des représentations sémantiques. Le fait que la grammaire capture les paraphrases en associant différentes expressions de langage naturel ayant la même signification noyau, est spécialement intéressant dans le contexte de l'apprentissage des langues. Généralement, les professeurs éditent manuellement des exercices et leurs solutions, puis les classent suivant leur degré de difficulté et le niveau attendu de l'apprenant. L'approche que nous proposons, appelée *GramEx*, permet potentiellement la (semi-)automatisation de l'ensemble du processus. Premièrement, en raison de la sous-spécification des entrées et de la

[1] `http://hotpot.uvic.ca/`

génération de paraphrases; plusieurs réalisations sont possibles à partir d'une seule entrée. Comme nous le montrons dans le chapitre 4, à partir d'une signification principale, plusieurs paraphrases sont générées et peuvent, à leur tour, chacune être utilisées pour construire plusieurs exercices différents. Dans ce sens, notre approche décharge le professeur de langues d'écrire manuellement chaque alternative ou de ré-écrire manuellement une phrase donnée à utiliser dans un autre type d'exercice. Deuxièmement, les riches informations linguistiques associées au texte généré peuvent être exploitées pour la génération automatique d'activités d'apprentissage. Ici, nous montrons comment les exercices de grammaire de type à textes-à-trous, de mots mélangés : reconstitution ou de reformulation de phrases peuvent être automatiquement créés. Troisièmement, la génération d'exercices à partir de la génération automatique de phrases permet potentiellement la classification automatique des exercices générés et leur ordonnancement dans une séquence pédagogique. Par exemple, les constructions grammaticales pourraient être élaborées suivant différents degrés de difficulté. A cette fin, *GramEx* peut être intégré dans une application telle que I-FLEG (Interactive French Learning Game, [Amoia *et al.*, 2012]) comme cela va être discuté dans la section 5.2. Dans I-FLEG, les interactions de l'apprenant avec le jeu sont stockées dans une base de données et fournissent des informations détaillées concernant, pour chaque exercice, les items résolus par l'apprenant. Ces informations peuvent être exploitées, par exemple, pour fournir automatiquement un entraînement spécifique sur les points de grammaire que l'apprenant doit améliorer.

Les livres d'apprentissage de langues incluent en général des exercices de grammaire. Par exemple, le livre en ligne *Tex's French Grammar* [2] for instance, includes at the end of each lecture, a set of grammar exercises which target a specific pedagogical goal such as *learning the plural form of nouns* or *learning the placement of adjectives* inclut, à la fin de chaque unité, un ensemble d'exercices de grammaire visant un but pédagogique spécifique tel que *l'apprentissage de la forme plurielle des noms* ou *l'apprentissage de l'ordre des adjectifs*. La Figure 1 montre les exercices se trouvant à la fin de l'unité sur la formation du pluriel des noms. Comme on peut le voir sur cette figure, ces exercices diffèrent notablement des activités d'apprentissage avancées qui cherchent à familiariser l'apprenant avec des phrases "couramment employées". Pour permettre l'apprentissage *in situ*, ce dernier type d'activité confronte l'apprenant

[2] *Tex's French Grammar* `http://www.laits.utexas.edu/tex/` is an online pedagogical reference grammar that combines explanations with surreal dialogues and cartoon images. Tex's French Grammar is arranged like many other traditional reference grammars with the parts of speech (nouns, verbs, etc.) used to categorize specific grammar items (gender of nouns, irregular verbs). Individual grammar items are carefully explained in English, then exemplified in a dialogue, and finally tested in self-correcting, Fill-In-the-Blank exercises.

avec des phrases extraites du Web ou des documents existants, l'exposant ainsi à des syntaxes potentiellement complexes et du vocabulaire varié. En revanche, les livres d'apprentissage de langues ont généralement pour but de faciliter l'acquisition d'un point de grammaire spécifique en confrontant l'apprenant avec des exercices construits à partir de phrases courtes et de vocabulaire restreint.

Give the plural form of the noun indicated in parentheses. Pay attention to both the article and the noun.

1. Bette aime _____ . (le bijou)
2. Fiona aime _____ . (le cheval)
3. Joe-Bob aime _____ américaines. (la bière)
4. Tex n'aime pas _____ . (le choix)
5. Joe-Bob n'aime pas _____ difficiles. (le cours)
6. Tammy n'aime pas _____ . (l'hôpital)
7. Eduard aime _____. (le tableau)
8. Bette aime _____ de Tex. (l'oeil)
9. Tex aime _____ français. (le poëte)
10. Corey aime _____ fraîches. (la boisson)
11. Tammy aime _____ américains. (le campus)
12. Corey n'aime pas _____ . (l'examen)

Figure 1: Exercices de grammaire issus du livre d'apprentissage de langues *Tex's French Grammar*.

Comme nous l'avons discuté dans les précédents paragraphes, la plupart des travaux existants sur la génération d'exercices de grammaire s'est concentrée sur la création automatique d'exercices du premier type, c'est à dire des exercices dans lesquels les phrases sources sont extraites à partir de corpus existants. Dans cette thèse, nous présentons une architecture qui vise les exercices de deuxième type, c'est à dire les exercices de grammaire dans lesquels la syntaxe et le vocabulaire sont fortement contrôlés.

Nous utilisons un réalisateur de surface utilisant une grammaire pour produire des phrases qui, suite au processus de génération, sont associées à de riches informations linguistiques. Nous définissons un mécanisme basé sur les informations linguistiques pour sélectionner les phrases appropriées. Plus précisément, nous nous intéressons aux exercices de deux types : d'une part, des exercices générés à partir d'une seule phrase tels que les exercices de type textes-à-trous ou mots mélangés, .d'autre part, des exercices de reformulation ou transformations de phrases qui requièrent un couple de phrases.

La production (semi-)automatique d'activités à partir du Web ou de documents existants a contribué à la création à grande échelle d'exercices tels que les exercices de type questionnaires à choix multiples ou textes-à-trous. En majorité, ces approches associent des annotations syntactiques et morpho-syntactiques avec les phrases collectées grâce à l'utilisation des techniques d'analyse de phrases, d'étiquetage, d'étiquetage morpho-syntactique et de segmentation de phrases . Cependant, la génération au-

tomatique d'exercices basés sur la transformation de phrases requiert des techniques d'analyse linguistique plus profondes et a reçu peu ou pas d'attention.

Considérons, par exemple, le cas de la production automatique de paires question (Q) et réponse (S) pour l'exercice suivant:

(2) Ré-écrire les phrases suivantes en utilisant la **voix passive**.

1. (Q) *C'est Tex qui donne le livre a Tammy.*
   It is Tex who gives the book to Tammy
2. (S) *C'est par Tex que le livre est donné a Tammy.*
   It is by Tex that the book is given to Tammy

Pour produire automatiquement la solution (S), nous avons besoin de générer une phrase qui contient le même sens et qui est exprimée en voix passive. En outre, il est également nécessaire que les autres caractéristiques syntactiques et morpho-syntactiques (par exemple le temps du verbe et la thématisation) soient maintenues le plus proche possible de la phrase originelle dans la question (Q). Notre grammaire F-TAG fournit des informations linguistiques détaillées (contenu sémantique, syntactique et morpho-syntactique) nécessaires pour identifier les paires de phrases qui sont liées par une transformation syntactique. En particulier, les arbres de dérivation de la grammaire FB-TAG constituent un bon niveau de représentation pour l'analyse de transformation syntactique parce qu'ils capturent à la fois les contraintes formelles et de contenus gouvernant les transformations syntactiques. Les mots pleins et les fonctions grammaticales étiquetant les nœuds des arbres permettent de vérifier que deux phrases se trouvent dans la relation sémantique appropriée (p.ex. contenu complètement identique ou contenu identique modulo des changements locaux). De plus, les propriétés syntactiques étiquetant ces nœuds (les noms des arbres élémentaires FBL-TAG mais également les informations linguistiques additionnelles fournis par le générateur) permettent d'assurer qu'elles se trouvent dans la relation syntactique appropriée.

Contributions principales

Les contributions de cette thèse sont les suivantes:

- Un nouvel algorithme pour la réalisation de surface basée sur une grammaire FB-TAG. Cet algorithme repose sur un encodage FB-RTG des arbres de dérivation de la grammaire FB-LTAG et incorpore des techniques d'optimisation

variées: partage et compression des structures intermédiaires, indexation basée sur les indices sémantiques et filtrage des structures intermédiaires incomplètes

- Une approche basée sur la génération automatique de langage naturel pour la génération automatique d'exercices similaires à ceux présents dans les livres d'apprentissage des langues. Nous exploitons les représentations sémantiques d'entrée sous-spécifiée ainsi que le pouvoir paraphrastique de la grammaire *SemTAG* pour produire des exercices grammaticaux. Les riches informations linguistiques associées aux phrases générées permettent la création (semi-)automatique d'exercices de grammaire.

- Une nouvelle approche pour la génération d'exercices de reformulation. Nous utilisons l'information contenue dans les arbres de dérivation de FB-LTAG pour identifier les paires de phrases qui sont liées par une transformation syntactique.

Structure de la thèse

Dans ce chapitre, nous introduisons les problématiques de recherche dont traite la présente thèse. Dans ce qui suit, nous résumons le contenu des chapitres restant de la thèse.

Chapter 2: Background and related work. Dans ce chapitre, nous passons brièvement en revue les concepts majeurs pour les deux thématiques couvertes par les travaux de cette thèse, à savoir la génération de phrases (Section 2.1) et l'apprentissage des langues assisté par ordinateur (Section 2.3), le but de cette démarche étant de situer nos travaux dans ces larges domaines. Pour la génération automatique de phrases, nous discutons des problèmes de complexité dans la réalisation de surface à partir de sémantiques plates. Pour CALL, nous discutons des travaux connexes qui motivent nos travaux sur l'application des techniques de génération automatique de phrases pour la génération d'exercices grammaticaux. Enfin, nous décrivons *SemTAG*, la grammaire utilisée par notre générateur, dans la Section 2.2 et mettons en avant les définitions et caractéristiques de cette grammaire qui sont pertinentes pour son utilisation dans le cadre de cette thèse.

1 Optimisation du module de réalisation de surface

La réalisation de surface à partir de sémantiques plates est exponentielle par rapport à la taille de l'entrée (nombre de prédicats) dans le pire cas, Les causes majeures

de cette complexité sont le manque d'ordre des informations et l'ambiguïté lexicale (cf Chapitre 2). Pour optimiser la réalisation de surface basée sur les grammaire d'arbres adjoints (TAG), nous proposons une approche, basée sur un encodage dans une grammaire d'arbres réguliers (FB-RTG, [Schmitz and Le Roux, 2008]) des arbres de dérivation de la grammaire FB-TAG, qui est inspirée de [Koller and Striegnitz, 2002].

Notre hypothèse repose sur le fait que l'utilisation de cet encodage permet de simplifier et d'optimiser la réalisation de surface basée sur la grammaire TAG. Nous commençons par décrire l'approche de Koller et Striegnitz, nous donnons ensuite les principes de notre approche et enfin nous présentons la traduction de FB-TAG vers FB-RTG de Schmitz et Le Roux dans la Section 3.1. Nous présentons un nouvel algorithme pour la réalisation de surface TAG basé sur l'encodage [Gardent and Perez-Beltrachini, 2010; Gardent *et al.*, 2011a], appelé RTGen, dans la Section 3.2. Nous réalisons une évaluation comparative en utilisant des cas de suites de tests graduées. [Gardent *et al.*, 2010; Gardent *et al.*, 2011a]. Nous discutons les résultats dans la Section 3.3. Dans la Section 3.4, nous comparons notre approche avec les travaux en lien avec l'optimisation de la réalisation de surface. Nous présentons nos conclusions dans la Section 3.5.

2 Génération automatique de texte pour l'apprentissage des langues

La génération automatique de matériel et d'activités pour l'apprentissage des langues a été abordée par les techniques d'analyse de phrases. Dans ce Chapitre, nous explorons une autre alternative basée sur les techniques NLG. Nous exploitons un concept clé de la génération à partir de représentations sémantiques sous-spécifiées avec une grammaire paraphrastique, à savoir la génération de paraphrases alternatives permise par la grammaire et la possibilité de choisir parmi celles-ci. En outre, notre approche de génération basée sur une grammaire produit du texte et, dans le même temps, ce texte est associé avec une représentation linguistique détaillée. Nous développons un logiciel appelée *GramEx*, pour la génération d'exercices de grammaire. Dans la Section 4.1, nous décrivons le type d'activité pour l'apprentissage que nous générons. La Section 4.2.1 décrit l'approche pour l'obtention de textes qui intègre les contraintes liées aux objectifs pédagogiques donnés ainsi et aux connaissances de l'apprenant. Une fois que le texte approprié a été produit, il peut être exploité pour construire des exercices de grammaire de plusieurs types. Dans la Section 4.3, Nous montrons comment les exercices de grammaire de type à textes-

à-trous, de mots mélangés dérivés à partir du texte généré [Perez-Beltrachini *et al.*, 2012]. Nous effectuons une évaluation qui montre l'*utilité* de l'approche au regard de la production d'exercices. En premier lieu, nous mesurons la variabilité, c'est à dire comment le degré de variation de phrases issues du processus de génération permet la création d'exercices variés. Deuxièmement, nous mesurons la productivité, c'est à dire, dans quelle mesure le la même phrase générée peut-elle être utilisée pour créer différent types d'exercices et combien d'exercices sont créés a partir d'une entrée donnée. Nous évaluons également l'*exactitude* qui permet de vérifier si les exercices générés son corrects et significatifs. Dans la Section 4.4, nous démontrons que l'approche permet la génération automatique d'exercices de reformulation de phrases ([Gardent and Perez-Beltrachini, 2012]). Nous résumons les travaux conduits et concluons dans la Section 4.6.

3 Conclusions

Nous dressons nos conclusions sur notre thématique basée sur notre propre réalisateur de surface et ses applications dans le cadre de l'apprentissage des langues (Section 5.1). Dans la Section 5.2, nous approfondissons les pistes de travail futur sur cette thématique.

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is about using Natural Language Generation (NLG) techniques in Computer-Assisted Language Learning (CALL). We show in particular how a grammar-based Surface Realiser (SR) can be usefully exploited to automate the generation of grammar exercises for language learning. The surface realiser uses a wide-coverage reversible grammar namely *SemTAG*, a Feature-Based Tree Adjoining Grammar (FB-TAG) equipped with a unification-based compositional semantics.

The thesis falls into two parts:

- In the first part, we examine the task of generating sentences from semantic formulae and propose an optimised algorithm that supports the generation of longer sentences given a large scale grammar and lexicon.

- In the second part, we explore how our *SemTAG*-based surface realiser can be exploited for the generation of grammar exercices whose syntax and vocabulary can be controlled. We propose an approach that takes advantage of the particular features of the underlying grammatical framework and the realiser. First, the grammar constitutes a precise and rich linguistic resource describing natural language expressions. This permits the generation of text material that satisfies certain syntactic and morpho-syntactic constraints (for instance, those imposed by a pedagogical goal of *learning passive voice*). Moreover, the rich linguistic information associated with the generated text by our realiser permits further processing it to create exercise items of the type Fill-in-the-blank, Shuffle and Reformulation. Second, the underspecified input and thus the several output produced by our surface realiser make it possible to automatically obtain syntactic and morpho-syntactic varied text material, and in turn exercise items, from few input.

The goal of the NLG task is to produce understandable text in human language. This process is governed by a given communicative intention, based on some information source, and involves a series of steps or subtasks. Traditionally, these subtasks are thought to be organised in a sequence or pipeline and to deal with strategic decisions ("what to say") and tactical decisions ("how to say it"). Once the content or meaning to be expressed in natural language has been determined, the tactical component needs to make several choices such as the words and syntactic structures to be used to express that meaning in natural language. In particular, the surface realisation component of an NLG system, usually the last component in the pipeline, maps an abstract linguistic specification into a natural language expression. That is, it knows about the target language, for instance, its word order. There can be different levels of abstractions in the specification of the input to the realiser. For instance, the input could be a dependency tree where syntactic roles have been specified as well as function words. Depending on the degree of specification of its input, the realiser could be deterministic or produce several output taking various decisions about how to say things. In this thesis, we assume a semantic input (e.g a logical form), and more specifically, a Minimal Recursion Semantic (MRS[3]) semantic formula. Given the MRS shown in (3a), the task of the surface realiser is to output sentences such as (3b-c).

(3) a. $\{l_0 : named(t,\ Tex),\ l_0 : indiv(t,\ m,\ sg),\ qeq(TR,\ l_0),\ l_1 : properq(t,\ TR,\ TS),$
$l_2 : le(u,\ CR,\ CS),\ qeq(CR,\ l_3),\ l_3 : universite(u),\ l_3 : indiv(u,\ f,\ sg),$
$l_4 : travailler(e,\ t,\ u),\ l_4 : event(e,\ pres,\ indet,\ ind)\}$

b. *Tex travaille à l'université.* (Tex works at the university)

c. *C'est Tex qui travaille à l'université.* (It is Tex who works at the university)

surface realisation optimisation

Surface realisation from flat semantics (i.e. a bag of predications as illustrated in example (3a)) is a computationally expensive task (Brew (1992) and Koller and Striegnitz (2002) provide NP-completeness proofs). Various optimisation techniques have been proposed to help improving runtimes in practice. Our goal is the optimisation of FB-TAG based surface realisation. To this end, we follow the idea of using TAG derivation trees for generation from [Koller and Striegnitz, 2002]. We depart from this approach however in that we rely on a well defined translation from FB-TAG to an FB-RTG (Feature-Based Regular Tree Grammar, [Schmitz and Le Roux, 2008]) to describe the derivation tree language of the FB-TAG. This translation carries over all semantic, syntactic and morpho-syntactic information from the original

[3]MRS are flat underspecified semantic representations ([Copestake *et al.*, 2005]), i.e. they are flattened and scope underspecified representations of First Oder Logic (FOL) formulae. We discuss this type of semantic representations in detail in sections 2.1.1 and 2.2.4

grammar having, thus, important consequences that distinguish our approach from previous ones. By preserving all linguistic information, the FB-RTG encoding preserves the syntax/semantics interface and provides an exact grammar of FB-TAG derivation trees. We provide an FB-RTG based surface realisation algorithm, namely RTGen, which integrates various techniques to improve surface realisation.

NLG technology has been used *inter alia* to generate reports (for instance, to generate text from a database of measurements from some measuring device), to generate descriptions from an underlying knowledge base and to map the output of a dialogue manager to a natural language expression. The domains of application have widely varied too: medical, weather forecasting, instructional leaflets, verbalizing instructions in virtual environments, among others. Moreover, other types of Natural Language Processing (NLP) tasks such as text summarization or simplification and question generation may also involve a final re-generation step. To a lesser extent, NLG techniques have also been used within the context of Intelligent Computer-Assisted Language Learning (ICALL).

A variety of work in NLP and ICALL has been carried out along the past years. Mostly, NLP techniques have been used to support the authoring of learning content and learning activities or to evaluate learner input and generate appropriate feedback. For instance, ALICE-chan ([Levin and Evans, 1995]) is an intelligent language tutor for Japanese instruction that uses Lexical Functional Grammar (LFG) based parsing to assist instructors in creating exercises and to evaluate learner answers to those exercices. It provides an interface for exercise authoring where instructors can create exercises by entering text corresponding to the background, the question and the answer of the exercises. The answer is analysed by the NLP modules which produce a feature structure summarizing morpho-syntactic and syntactic features that will be used later on to evaluate learner input (which is analysed in a similar way). Another tutoring system is TAGARELA ([Amaral and Meurers, 2011]). It includes workbook style activities: reading and listening comprehension, picture description, rephrasing, Fill-in-the-blank and vocabulary exercises. Different NLP tools (e.g. tokenizer and parser) are deployed in its architecture. TAGARELA focuses on processing learner input and providing appropriate feedback making use of expert models (knowledge about the language), activity models and learner models.

Within the range of ICALL applications, there are some systems that embody writing aids such as ICICLE ([Michaud *et al.*, 2000]) or reading assistants such as CALLE ([Rypa and Feuerman, 1995]) which rely on parsing techniques. ICICLE uses parsing to analyse learners' input, whereas CALLE uses parsing to analyse documents selected by the learner to provide information about the linguistic con-

structions present in the text. In different ways, both aim at emphasizing awareness and learning of the grammatical constructions in the target language. Two other systems whose major aim is to promote linguistic awareness are WERTI ([Meurers *et al.*, 2010]), a so-called text enhancement application, using NLP tools to recognise and highlight different grammatical features in selected Web documents; and VISL ([Bick, 2005]), a visual interactive syntax learning tool, using also NLP tools for analysis.

Some work has specially concentrated on the automatic authoring of language learning exercise and test items ([Mitkov *et al.*, 2006; Heilman and Eskenazi, 2007; Karamanis *et al.*, 2006; Chao-Lin *et al.*, 2005; Coniam, 1997; Sumita *et al.*, 2005; Simon *et al.*, 2010; Lin *et al.*, 2007; Lee and Seneff, 2007]). In particular, some proposals target the production of grammar exercices (e.g. [Aldabe *et al.*, 2006; Chen *et al.*, 2006]). In general, these approaches rely on machine learning techniques and generate advanced learning activities.

Within CALL there exist authoring tools such as *Hot Potatoes*[4] ([Winke and MacGregor, 2001]) which do not use NLP techniques. They are called template based authoring tools because they provide a set of template activities that the language teacher can use to create exercises. However, the content for each exercise, that is, the source text, the expected solution(s) and the feedback, is manually entered by the language teacher.

In sum, there exist CALL applications where the language learning material is edited by hand or ICALL applications in which most of the work on NLP devoted to the (semi-)automatic creation of learning material is based on text analysis. In the latter case, the textual content used to create learning activities is either provided by the language teachers or gathered automatically from the Web.

In this thesis, we argue that NLG is a natural candidate for the (semi-)automatic generation of language learning material. We exploit an FB-TAG wide-coverage paraphrastic grammar which provides a rich linguistic description of natural language associating natural language expressions with syntax and semantics. The fact that the grammar captures paraphrases by associating different natural language expressions with the same underlying core meaning is specially attractive in the context of language learning. Usually, teachers manually edit exercises and their solutions, and classify them according to the degree of difficulty or the expected learner level. The approach we propose, called *GramEx*, potentially supports the (semi-)automation of the whole process. First, due to input underspecification and paraphrase generation, from one input several realisations are possible. As we show

[4] `http://hotpot.uvic.ca/`

in Chapter 4, from a given core meaning several paraphrases are generated which, in turn, might be used to build several different exercices. In this way, our approach releases the language teacher from manually writing each alternative or from manually re-writing a given sentence to be used in another exercise type. Second, the rich linguistic information associated with the generated text material can be exploited for the automatic generation of learning activities. Here, we show how Fill-in-the-blank, Shuffle and Reformulation grammar exercises can be automatically created. In this way, the language teacher does not need to manually modify the text or to enter the solutions. Third, the NLG-based exercise generation approach potentially enables the automatic classification of the generated exercices for instructional sequencing. For instance, the grammar constructions could be mapped to different levels of difficulty. Further, as will be discussed in the future work section 5.2, *GramEx* has been integrated in the I-FLEG application (Interactive French Learning Game, [Amoia *et al.*, 2012]), a serious game for practicing grammar exercices in French. In I-FLEG, the learner interactions are stored in a database and provide detailed information about each exercise item solved by the learner. This information could be exploited, for instance, to automatically provide training in the grammar points that a given learner needs to reinforce.

textbook-style exercises

Textbooks for language learning generally include grammar exercises. *Tex's French Grammar* [5] for instance, includes at the end of each lecture, a set of grammar exercises which target a specific pedagogical goal such as *learning the plural form of nouns* or *learning the placement of adjectives*. Figure 1.1 shows the exercises provided by this book at the end of the lecture on the plural formation of nouns. As exemplified in this figure, these exercises markedly differ from more advanced learning activities which seek to familiarise the learner with "real world sentences". To support *in situ* learning, this latter type of activity presents the learner with sentences drawn from the Web or from existing documents thereby exposing her to a potentially complex syntax and to a diverse vocabulary. In contrast, textbook grammar exercises usually aim to facilitate the acquisition of a specific grammar point by presenting the learner with exercises made up of short sentences involving a restricted vocabulary.

As we argued in previous paragraphs, most existing work on the generation of

[5] *Tex's French Grammar* `http://www.laits.utexas.edu/tex/` is an online pedagogical reference grammar that combines explanations with surreal dialogues and cartoon images. Tex's French Grammar is arranged like many other traditional reference grammars with the parts of speech (nouns, verbs, etc.) used to categorise specific grammar items (gender of nouns, irregular verbs). Individual grammar items are carefully explained in English, then exemplified in a dialogue, and finally tested in self-correcting, Fill-In-the-Blank exercises.

Give the plural form of the noun indicated in parentheses. Pay attention to both the article and the noun.

1. Bette aime _____ . (le bijou)
2. Fiona aime _____ . (le cheval)
3. Joe-Bob aime _____ américaines. (la bière)
4. Tex n'aime pas _____ . (le choix)
5. Joe-Bob n'aime pas _____ difficiles. (le cours)
6. Tammy n'aime pas _____ . (l'hôpital)
7. Eduard aime _____. (le tableau)
8. Bette aime _____ de Tex. (l'oeil)
9. Tex aime _____ français. (le poëte)
10. Corey aime _____ fraîches. (la boisson)
11. Tammy aime _____ américains. (le campus)
12. Corey n'aime pas _____ . (l'examen)

Figure 1.1: Grammar exercises from the *Tex's French Grammar* textbook

grammar exercises has concentrated on the automatic creation of the first type of exercises i.e., exercises whose source sentences are extracted from an existing corpus. Here, we present a framework which addresses the generation of the second type of grammar exercises used for language learning i.e., grammar exercises whose syntax and vocabulary are strongly controlled.

We use our grammar-based surface realiser to produce sentences which, as a result of the generation process, are associated with rich linguistic information. We define a mechanism to select appropriate sentences based on this linguistic information. We show how these sentences can be further processed to generate grammar exercises. More precisely, we focus on exercises of two types. The first type, includes those exercises that are generated from a single selected sentence such as the Fill-in-the-blank and Shuffle exercises. The second type, is the Reformulation or transformation exercise type which requires a pair of sentences.

transformation-based grammar exercises

The (semi-)automatic production of activities from the Web or existing documents has contributed to the large scale creation of exercises such as multiple-choice or Fill-in-the-blank. Mostly, these approaches associate syntactic and morpho-syntactic annotations with the collected sentences using parsing, part-of-speech tagging and chunking techniques. However, the automatic generation of transformation-based exercise types requires deeper linguistic processing and has received little or no attention. Consider, for instance, the case of automatically producing question (Q) and expected answer (S) pairs for the following active/passive Reformulation exercise:

(4) Rewrite the sentences using **passive voice**.

1. (Q) *C'est Tex qui donne le livre a Tammy.*
   It is Tex who gives the book to Tammy

2. (S) *C'est par Tex que le livre est donné a Tammy.*
   It is by Tex that the book is given to Tammy

To automatically produce the solution (S), we need to generate a sentence that bears the same core meaning and is in the passive voice, but also the other syntactic and morpho-syntactic features (e.g. topicalization and tense) should be maintained as close as possible to the original sentence in the question (Q). Our FB-TAG grammar provides the detailed linguistic information (i.e. semantic content and syntactic and morpho-syntactic information) necessary to identify sentence pairs that are related by a syntactic transformation. In particular, the derivation trees of the FB-TAG grammar provide a good level of representation for analysing syntactic transformations as they capture both the formal and the content constraints governing transformations. The content words and the grammatical functions labelling the tree nodes permit checking that the two sentences stand in the appropriate semantic relation (i.e., fully identical content or identical content modulo some local change). Further, the syntactic properties labelling these nodes (FB-LTAG elementary tree names but also some additional information provided by our generator) permit ensuring that they stand in the appropriate syntactic relation.

Main contributions

The contributions of this thesis are the following.

- A new algorithm for FB-TAG based surface realisation. This algorithm relies on an FB-RTG encoding of FB-TAG derivation trees and incorporates various optimisation techniques: packing, sharing, indexing based on semantic arguments and filtering of intermediate incomplete structures.

- An NLG-based approach for the automatic generation of textbook-style exercises. We exploit the underspecified input semantic representations and the paraphrastic power of the *SemTAG* grammar to produce text material. The rich linguistic information associated with the generated sentences permits the (semi-)automatic creation of grammar exercises.

- A novel approach for the generation of transformation-based grammar exercises. We use FB-TAG derivational information to identify pairs of sentences that are related by a syntactic transformation.

Road map of the thesis

In this chapter, we introduced the research issues this thesis addresses. In what follows, we summarise the content of the remaining chapters of the thesis.

Chapter 2: Background and related work. In this chapter, we briefly survey the major concepts underlying the two research trends adressed in this thesis, namely natural language generation (Section 2.1) and computer-assisted language learning (Section 2.3) and we situate our work within these broad areas. For NLG, we discuss the complexity issues in surface realisation from flat semantics that motivate our research on surface realisation optimisation. For CALL, we discuss related work that motivates our research on applying NLG techniques for the generation of language learning material. Finally, we describe SemTAG, our underlying grammar framework, in (Section 2.2). In doing this, we aim at emphasizing those definitions or features of the underlying grammatical formalism and our specific grammar implementation that are needed for the discussion of the presented approaches.

Chapter 3: Optimising surface realisation. Surface realisation from flat semantic is exponential in the size of the input (number of predications) in the worst case. The major causes for this complexity are the lack of ordering information and the lexical ambiguity (cf. Chapter 2). To optimise TAG-based surface realisation, we propose an approach based on an FB-RTG ([Schmitz and Le Roux, 2008]) encoding of FB-TAG derivation trees that is inspired from [Koller and Striegnitz, 2002]. Our hypothesis is that using this encoding permits simplifying and optimising FB-TAG-based surface realisation. We begin by describing Koller and Striegnitz's approach, giving the intuitions about our approach, and presenting Schmitz and Le Roux's FB-TAG to FB-RTG translation in Section 3.1. We present a new algorithm for FB-TAG-based surface realisation based on this encoding ([Gardent and Perez-Beltrachini, 2010; Gardent *et al.*, 2011a]), namely RTGen, in Section 3.2. We carried out a comparative evaluation using automatically built graduated test-suites ([Gardent *et al.*, 2010; Gardent *et al.*, 2011a]). We discuss the results in Section 3.3. In Section 3.4, we compare our approach with related work on surface realisation optimisation. We conclude in Section 3.5.

Chapter 4: Natural language generation for language learning. The automatic generation of learning content and learning activities has mostly been addressed by using text analysis and machine learning techniques. In this chapter, we explore an alternative approach which uses NLG techniques. We exploit a key

concept of the generation from underspecified input semantics with a paraphrastic grammar, that is, the generation of all alternative paraphrases licensed by the grammar and the possibility of choosing among them. Moreover, our grammar-based generation approach generates sentences and XFthe generated sentences are associated with rich linguistic information produced by the generation process. We develop a framework, namely *GramEx*, for the generation of grammar exercises. In Section 4.1, we describe the type of learning activities we generate. Section 4.2.1 describes the approach for the generation of text material that supports pedagogical goals and learner knowledge constraints. Once the appropriate text material has been selected, we can exploit it to build different types of grammar exercises. In Section 4.3, we show how Fill-in-the-blank and Shuffle exercises can be derived from generated text ([Perez-Beltrachini *et al.*, 2012]). We carried out an evaluation that shows the *usefulness* of the approach in terms of exercise production. First, we measure variability, that is, to what extent the degree of variation in the output of the generation process permits the generation of varied exercises. Second, we measure productivity, that is, to what extent the same generated sentence serves to create different exercises as well as how many exercises can be produced from a given input. We also evaluate *correctness*, that is, whether the generated exercises are most of the time meaningful and correct. In Section 4.4, we show that the generation approach nicely supports the automatic creation of sentence reformulation type of exercises ([Gardent and Perez-Beltrachini, 2012]). We summarise the work carried out and conclude in Section 4.6.

Chapter 5: Conclusions. We draw our conclusions on our particular surface realisation task and its application to language learning (Section 5.1). In Section 5.2, we give pointers for further research.

Chapter 2

Background and related work

Contents

As we discussed in Chapter 1, one of our driving research question is "how sentence generation from underspecified logical for can optimised" and the other is "how NLG techniques can be exploited for the generation of language learning material". In this chapter, we introduce those concepts that we will use throughout the rest of the thesis and describe related work on computer assisted language learning motivating the research in this thesis.

First, we summarise the natural language generation task and describe the complexity issues related to surface realisation from flat semantics (Section 2.1). In Section 2.2, we describe the *SemTAG* grammatical framework in which our surface realiser is based on. Then, in Section 2.3, we briefly give an overview of the ICALL to situate our work and proceed to discuss related work.

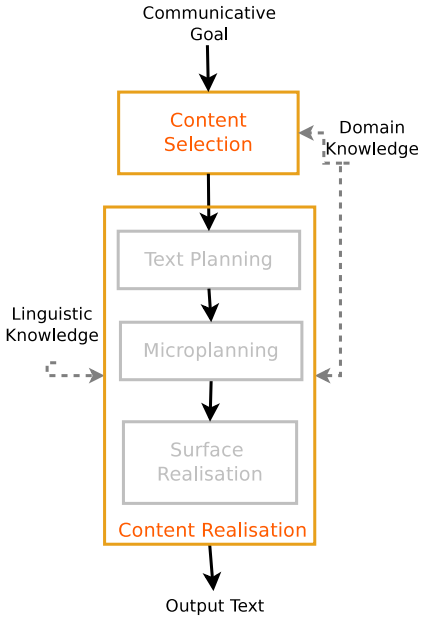


Figure 2.1: NLG pipeline architecture

2.1 Natural Language Generation

The tasks carried out by a natural language generator divide in (i) the selection of the content to be communicated and (ii) decisions about how that content should be expressed in the target natural language ([McKeown, 1992]). Traditionally, these tasks are thought to be organised in a *pipeline architecture* ([Reiter, 1994]). A picture of this pipeline is shown in Figure 2.1.

In the first stage, the **content selection** module selects the content to be communicated. In the second stage, **content realisation**, decisions regarding the text structure (*text planning*) and the sentence plan (*microplanning*) are taken. Sentence planning involves decisions about how the content is structured into sentences (aggregation), the selection of words and syntactic constructions (lexicalization) and how to refer to entities (referring expression generation). The last module, the *surface realiser*, based on linguistic knowledge about the target language (e.g. grammar and lexicon) transforms an abstract representation into a syntactically and morphologically correct text. However, in some NLG approaches, these tasks are carried out together rather than one after the other (for instance, Koller and Stone's (2007) approach interleaves the generation of referring expressions and surface realisation) or in a different order (e.g. in [Banik *et al.*, 2012] the referring expression generation task is carried out after surface realisation).

The type of input the surface realiser receives also marks a different distribution

of labour between the surface realiser and previous tasks (if we think of a pipeline architecture). For instance, if lexicalization decisions have not been taken by the microplanner it will be up to the surface realiser to make lexical choices. While a fully specified input determines the output sentence, with an underspecified input the surface realiser can either output all possible realisations or make the choice of the most appropriate realisation (or set of realisations).

Our work in this thesis focuses on the content realisation stage. More precisely, it focuses on a surface realiser which, given an underspecified input, produces all possible realisations licensed by a grammar. We exploit this feature for the generation of language learning exercises. We define a selection mechanism on top of the realisation step for choosing appropriate sentences for exercise generation (e.g. sentences satisfying a set of syntactic and morpho-syntactic constraints for a specific pedagogical goal).

In what follows, we give a brief overview of surface realisation and discuss the issues that arise in the type of realiser we work with.

Surface realisation

The surface realiser maps a text specification into a natural language expression, hence, it requires linguistic knowledge about the target language. There are different linguistic theories and grammatical formalisms which have been used to provide a realiser with the required linguistic knowledge. Examples of functional theories of grammar include Systemic Functional Grammar (SFG, [Halliday, 1985]) and Functional Unification Grammar (SURGE, [Elhadad *et al.*, 1997]). Among the grammatical formalisms following the generative grammar approach to the study of syntax are Head-driven Phrase Structure Grammar (HPSG, [Pollard and Sag, 1988]), Combinatory categorial grammar (CCG, [Steedman, 2000a; Steedman, 2000b]), Lexical-Functional Grammar (LFG, [Kaplan and Bresnan, 1981]) and TAG (the grammatical formalism our generator is based on, cf. Section 2.2), all of them providing phrase structure constituency descriptions. Meaning-Text Theory (MTT, [Melcuk, 1988]) is another linguistic framework used in generation which describes language at different levels, from semantics to phonetics, providing a representational model for each level and a map from one level to the next; the syntactic level (Syntactic Structure (SyntS)) relies on a dependency grammar. Finally, Performance Grammar (PG, [Kempen and Harbusch, 2002]) is a psycho-linguistically motivated formalism describing natural language syntax in terms of phrase structure but at the same time modelling the syntactic processing phenomena encountered during language production (e.g. incrementality).

In parsing, the surface form is known and the goal is to build a syntactic structure. In generation, it is a meaning representation that has to be mapped into a surface form, and in doing this other questions arise, for instance, in which context is this meaning used? which possible constructions does the language offer to express this meaning? ([Reiter and Dale, 1997]). Therefore, surface realisers have been proposed which are based on *generation-oriented grammars* which allow for linguistic descriptions in linguistic dimensions (e.g. functional) other than form. For instance, KPLM/Nigel ([Matthiessen and Bateman, 1991]) relies on Systemic Functional Grammar (SFG, [Halliday, 1985]) focusing on characterising language in terms of function rather than structure. General purpose well known surface realisers which rely on a generation-oriented linguistic resources are (in addition to KPML) RealPRO (based on MTT), FUF/SURGE, NITROGEN and HALOGEN.

generation-oriented grammar

However, the idea of using the same grammar both for parsing and generation emerges with the hope of economising resources development efforts ([Reiter and Dale, 1997]). Reusing wide-coverage grammars developed for parsing is an attractive alternative. These grammars used both for parsing and generation, namely *reversible grammars*, are equipped with a compositional semantics and describe relations between meaning and form. With a reversible grammar, a parser constructs meaning representations for a given sentence, whereas a generator takes as input a meaning representation and builds those sentences that are associated by the grammar with the given meaning. A surface realiser using such reversible grammars is called a *reversible realiser*. The input to this type of realiser is, in general, less specified than the input for those realisers geared towards a unique result discussed in previous paragraphs. For instance, given an input meaning such as that in (5a) [6], among many others, the sentences in (5b-g) would be produced.

reversible grammar

(5) a. offer(tex, tammy, watch)
b. *Tex offers a watch to Tammy.*
c. *Tammy is offered a watch by Tex.*
d. *A watch is offered by Tex to Tammy.*
e. *A watch is offered to Tammy by Tex.*
f. *It is Tex that offers a watch to Tammy.*
g. .....

For our particular application of NLG for language learning, we draw on a surface realiser of the second type (i.e. one using a reversible grammar equipped with compositional semantics). A priori, these surface realisers are not outfitted with a

[6] In the form of "skeletal propositions" (cf. [Reiter and Dale, 1997])

mechanism for specifying choices among different natural language expressions (e.g. different surface forms for a given meaning representation). However, as we will see in Chapter 4 the grammatical framework (cf. Section 2.2.5) in which our surface realiser builds on permits specifying form choices.

In what follows, we turn to the discussion of a particular type of meaning representation, "flat semantic formula", and discuss the complexity issues arising for surface realisation from this representation. Our surface realiser is based on this semantic representation.

2.1.1 Surface realisation from flat semantics: complexity issues

Lexicalist grammars (i.e. grammars assuming that the information necessary to build sentence structure and meaning comes from lexical items) are typically associated with a semantic representation that can be seen as a bag of lexical predicates. Such grammatical frameworks consist of a set of lexical rules and simple operations which do not introduce meaning to combine them ([Whitelock, 1992]). Examples of such frameworks are Head-driven Phrase Structure Grammar (HPSG, [Pollard and Sag, 1988]) and the grammatical formalism we use in this thesis, namely Tree Adjoining Grammar ([Joshi *et al.*, 1975; Joshi, 1987]).

Two concrete flat semantic frameworks are Minimal Recursion Semantics (MRS, [Copestake *et al.*, 2001; Copestake *et al.*, 2005]) and L_U –underspecified logic– ([Gardent and Kallmeyer, 2003]). The latter builds on MRS ([Copestake *et al.*, 2001]) and the language proposed by Bos (1995) for underspecified semantic representations; and it is the semantic framework that we use in this thesis (in Section 2.2.4 we present it and describe how it is integrated in the TAG grammar).

The motivation behind using flat semantics is discussed in detail in [Copestake *et al.*, 2001; Gardent and Kallmeyer, 2003]. In brief, there are two main motives. One concerns the problem of logical form equivalence of Shieber (1993), that is, which of the syntactic variants of a logical form does the grammar is associated with? For instance[7], (6a) and (6b) are syntactically different formulae but equivalent in meaning. The second motive concerns the problem of determining a particular scope reading of a given sentence. For instance, (7b) and (7c) are two different readings for the sentence in (7a).

(6) a. $\lambda x[fierce(x) \wedge (black(x) \wedge cat(x))]$
b. $\lambda x[cat(x) \wedge (black(x) \wedge fierce(x))]$

(7) a. *Every dog chases a cat.*

[7] Example taken from [Copestake *et al.*, 2005].

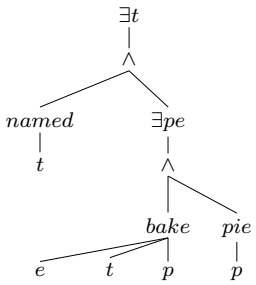


Figure 2.2: Tree for the semantic representation in (8b).

b. $\forall x(dog(x) \Rightarrow \exists y(cat(y) \ \wedge \ chases(x, y)))$
c. $\exists y(cat(y) \ \wedge \ \forall x(dog(x) \Rightarrow chases(x, y)))$

Conventional first-order representations of natural language meaning use recursion (or hierarchical structure) to model the meaning of some natural language expression as the meaning built from its sub-expressions. The example (8b) shows a meaning representation in first-order logic of the sentence in (8a). The subformula corresponding to the VP phrase *bakes a pie* (i.e. $\exists pe(bake(e,\ t,\ p) \wedge pie(p))$) models the fact that the VP is composed of a transitive verb and a direct object NP. The semantic representation in (8b) can be represented as a tree (see Figure 2.2).

(8) a. *Tex fait une tarte.* (Tex bakes a pie)
b. $\exists t(named(t, Tex) \wedge \exists pe(bake(e,\ t,\ p) \wedge pie(p)))$
c. $properq(t,\ named(t, Tex),\ exists(p,\ pie(p),\ bake(e,\ t,\ p)))$
d. $\{l_1 : named(t,\ Tex),\ l_0 : properq(t,\ l_1,\ l_2),\ l_2 : exists(p,\ l_3,\ l_4),$
$l_3 : pie(p),\ l_4 : bake(e,\ t,\ p)\}$

Furthermore, in linguistics the semantics of a quantifying determiner, i.e. *une*, according to the generalised quantifiers theory, is that of establishing a relation between the its nominal argument (*restriction* of the quantifier) and an external property or verb (*scope*). A propernoun expresses determination. Following Copestake *et al.*'s (2005) three-arguments syntax for generalised quantifiers, we can express the meaning of the sentence in (8a) as the semantic formula in (8c). The first argument of the quantifying determiner corresponds to the quantified variable, the second and third arguments correspond to its restriction and scope, respectively. We can give a tree representation of this formula as shown in Figure 2.3. The restriction and scope are represented as two daughters of the node corresponding to the determiner.

This tree structure can be flattened if the predicate (or set of predicates) at each node is identified with a "label", and then, the labels are used to refer to subformulae occurring as arguments of other predicates. For instance, in Figure 2.4, the labels

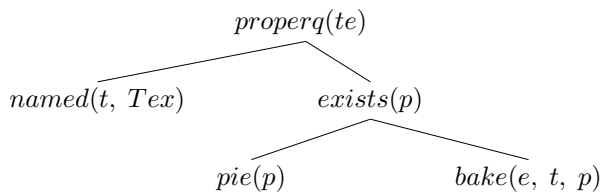


Figure 2.3: Tree for the semantic representation in (8c).

$$l_0 : properq(t,\ l_1,\ l_2)$$

$$l_1 : named(t,\ Tex) \qquad\qquad l_2 : exists(p,\ l_3,\ l_4)$$

$$l_3 : pie(p) \qquad\qquad l_4 : bake(e,\ t,\ p)$$

Figure 2.4: Disconnected tree representation with labelled predicates.

l_1 and l_2 label the predications corresponding to the restriction and scope of the generalised quantifier *properq* respectively ([Copestake *et al.*, 2005]). Furthermore, the obtained labelled predicates stand in an n-ary commutative and associative conjunction. As a result, they are considered as a set (or bag if there exist repeated elements) of predications. The formula in (8d) shows a flat semantic representation, more precisely, a MRS representation, of the sentence in (8a).

As is well-known, surface realisation from flat semantics is a computationally expensive task ([Brew, 1992; Kay, 1996; Koller and Striegnitz, 2002]). Proofs of its NP-completeness are given by Brew (1992) and Koller and Striegnitz (2002).

lack of order in the input

One first reason for the exponential complexity of surface realisation is the lack of ordering information. Contrary to parsing, in generation from flat semantics there are no string positions to guide the process (e.g. in chart parsing only adjacent edges are considered for combination). Supposing that each literal in the input semantics selects exactly one grammatical structure and that there are n literals, in a worst case configuration, there would be 2^n possible combinations among the selected structures. In practice, there are possible restrictions on structure combination. Flat semantic formulae can be used for "indexing", that is, for imposing some constraints on the combinations of selected structures. Most existing realisers impose the constraint that only constituents with non overlapping semantics and compatible indices can be combined (for instance in [Kay, 1996; Carroll *et al.*, 1999; Carroll and Oepen, 2005; White, 2004]).

modification

Because of these constraints, the exponential complexity manifests in modification ([Brew, 1992; Kay, 1996]). Given a set of k modifiers all modifying the same

structure, all possible intermediate structures will be constructed, i.e. 2^k. For instance, there are $2^3 = 8$ possible subsets of modifiers in *fierce little black cat*:

(9) *cat,*
fierce cat,
little cat,
black cat,
fierce little cat,
fierce black cat,
little black cat,
fierce little black cat

modifiers ordering

To this, it should be added that the order among modifiers is underspecified in flat semantic representations. Therefore, for each 2^k set of modifiers we have to consider the $k!$ possible orderings.

proliferation of intermediate incomplete structures

As Kay pointed out, the situation is serious if we consider that these 2^k intermediate incomplete structures might be further combined into larger phrases. For instance, towards building the sentence *The fierce little black cat sleeps*, the following incomplete sentences will be build: *The cat sleeps*, *The little cat sleeps*, and so on.

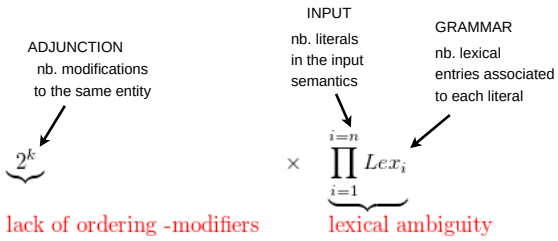
lexical ambiguity

A second reason for the exponential complexity of surface realisation is lexical ambiguity. In surface realisation from flat semantics, the literals in the input will probably select more than one grammatical structure (or lexical entry). Lexical ambiguity can come from different sources:

- Synonyms might be given the same semantic representation (e.g. (10)).
  (10) the case of *fast* and *quickly*
- Different uses of verbs: transitive and intransitive or the same lexical item with different functions: noun or verb (e.g. (11) and (12)) which might have overlapping semantic representations.
  (11) the case of *love* that is transitive and intransitive
  (12) the case of *place* that is a noun and a transitive verb
- If the grammar is paraphrastic i.e., associates several syntactic structures with the same semantics (e.g. (13)).
  (13) *John destroyed the castle quickly*
  The destruction of the castle by John was quick

So, given an input flat semantics with n literals, if Lex_i is the number of lexical entries associated with each literal l_i in the input semantics, then, the number of sets of lexical items covering the input semantics is: $\Pi_{i=1}^{i=n} Lex_i$. The two sources of

complexity (lexical ambiguity and lack of input ordering) interact by multiplying out so that the potential number of combinations of selected grammatical structures is:



Different solutions have been proposed to cope with the task complexity. Related work addressing modification includes: verifying inaccessible semantic indices ([Kay, 1996; Carroll and Oepen, 2005]), delayed insertion of modifiers ([Carroll *et al.*, 1999; Gardent and Kow, 2005]), chunking the input logical form ([White, 2004]), among others. Our approach relies on ambiguity packing and a mechanism to control proliferation of intermediate structures based on ideas from Kay (1996) and Carroll and Oepen (2005).

Carroll and Oepen's (2005) approach relies on local ambiguity packing to deal with lexical ambiguity, whereas the approaches proposed in [Gardent and Kow, 2006; Koller and Striegnitz, 2002; Bangalore and Rambow, 2000a] aim at reducing the initial search space by filtering out pointless constituent combinations.

In particular, unlike generating from a bag of predicates, Kanazawa (2007) shows that for generation from hierarchical and ordered input meaning representations (as well as for parsing from an input string) there exists a polynomial runtime algorithm. The particular generation task in [Kanazawa, 2007] concerns "exact generation", i.e. the input logical form is expected to exactly match that of the grammar. However, Kanazawa (2011) proposes a way to apply the framework to generation from underspecified input in which the different scope readings might be compactly represented in the input to the generation algorithm.

It is thus important to observe that flat semantic representations such as MRS ([Copestake *et al.*, 2005]) maintain hierarchical information, e.g. quantifiers' restriction or scope, as we have seen at the beginning of this section. That is, hierarchical information is merely represented in different way. For instance, Copestake (2009) shows that MRSs can be translated into semantic dependency graphs. These structural information contained in MRS representations could be exploited to guide the combination of grammar constituents during generation.

We will discuss related work on optimisation of surface realisation from flat semantics in more detail in Chapter 3.

2.2 The SemTAG grammar

In this section, we describe *SemTAG* the specific grammatical framework that we use for sentence generation, SemFraG the French version and SemXTAG the English one. *SemTAG* is a Feature Structures Based Lexicalised Tree Adjoining Grammar (FB-LTAG) augmented with a unification-based compositional semantics.

First, we will review the underlying grammatical formalism, FB-LTAG, and the integrated semantic representation language, L_U, and recap on those concepts (e.g. *derivation trees*) which are central in our approaches to surface realisation (Chapter 3) and also used in the generation of transformation-based grammar exercises (Chapter 4). Next, we discuss particular features related to our grammar implementation.

2.2.1 Tree Adjoining Grammar

Tree Adjoining Grammar (TAG, [Joshi and Schabes, 1997] –originally introduced in [Joshi *et al.*, 1975; Joshi, 1985]) is a grammatical formalism designed to describe natural languages.

Grammars describing natural languages structure need to account for linguistic dependencies, such as subcategorisation (eg. verb subcategorisation) and filler-gap dependencies (e.g. wh-movement). These dependencies can be at an unbounded distance, nested or crossed. As it is well known, natural languages can not be described by context free grammars. Some languages, e.g. Swiss-German ([Shieber, 1985]) and Dutch ([Bresnan *et al.*, 1982]), contain cross-serial dependencies.

TAG models these dependencies within a bounded structure, namely a *tree*, i.e. dependencies can be defined between nodes of a tree. Recursion is factorised from the basic unit describing dependencies and is implemented through a composition operation, namely *adjunction*, which permits rewriting a node of one tree with another tree. As corollary of this adjunction operation, locally defined dependencies might be "stretched" making them arbitrarily distant (unboundedness) and certain types of cross-dependencies can be accounted for (cf. [Joshi, 1985]). The latter is one of the formal properties that makes TAG fall in the class of *mildly-context sensitive grammars*.

Definition

Formally, a *tree adjoining grammar* is a quintuple $\langle \Sigma, N, I, A, S \rangle$ with Σ a set of terminals, N a set of non-terminals, I a finite set of **initial trees**, A a finite set of **auxiliary trees**, and S a distinguished non-terminal ($S \in N$). Initial trees are trees whose leaves are labelled with substitution nodes (marked with a downarrow: $\downarrow$) or

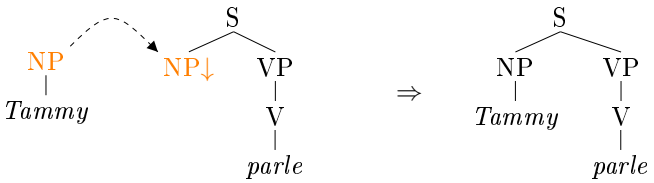


Figure 2.5: Example of substitution operation in TAG

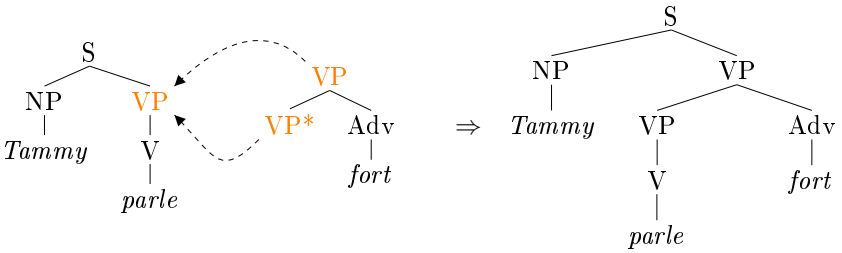


Figure 2.6: Example of adjunction operation in TAG

terminal categories. Auxiliary trees are distinguished by a foot node (marked with a star: $*$) whose category must be the same as that of the root node. In both, initial and auxiliary trees, internal nodes are labelled by non-terminal symbols. Trees in the set $I \cup A$ are called **elementary trees**. The tree obtained by composition of two elementary trees is called **derived tree**.

Two tree-composition operations are used to combine trees: **substitution** and **adjunction**. Substitution (Figure 2.5) replaces a leave non-terminal node X of an initial tree with another tree with root node X. The substituted tree should be derived from an initial tree. A tree with no substitution node is called a complete tree. Adjunction (Figure 2.6) inserts an X-type auxiliary tree into an initial or derived tree at a non-terminal node labelled by X (the non-terminal node should not be substitution node).

Adjunction constraints. The definition of the tree-composition operations given above states two constraints: the nodes where the operations take place should be labelled with the same non-terminal symbol and adjunction cannot take place at a substitution node. In addition, adjunction constraints allow to specify (linguistically motivated) restrictions stating, for instance, which auxiliary trees can adjoin into a given node. Thus, nodes in TAG elementary trees can be marked with a (i) *selective adjunction* constraint, $SA(T)$, stipulating that only trees in the set T, with $T \subseteq A$, can adjoin (note: adjunction is not compulsory), (ii) a *null adjunction* constraint, NA, disallowing any adjunction, or (iii) an *obligatory adjunction* constraint, $OA(T)$,

requiring that an auxiliary tree member of $T \subseteq A$ adjoins into the node.

In the next section, we introduce Lexicalised TAG (LTAG) and describe TAG's embedding in a feature structures based unification framework (Feature Based TAG (FB-TAG or FTAG)).

2.2.2 *Feature Structures Based Lexicalised* TAG

Lexicalised TAG

According to Schabes *et al.* (1988), a grammar is lexicalised if it consists of: (i) a finite set of structures each associated with a lexical item, called the **anchor** of the corresponding structure, and (ii) an operation or operations for composing these structures.

Lexicalised grammars associate lexical items with elementary structures describing their possible syntactic configurations. In practice, as shown in [Schabes *et al.*, 1988], parsing algorithms benefit from lexicalised grammars. The parsing process can be divided in two stages. In a first stage, the parser selects only those elementary structures that correspond to lexical items in the input string, viz. *sub-grammar selection*. In a second stage, the parser combines the selected structures. The combination step takes advantage of the fact that an elementary structure corresponds to a token in the input string[8] and therefore it can be used only once in a given parse. Moreover, since the sub-grammar used for parsing is selected according to the input string, non-local information might be used to guide the combination. These advantages also exist in generation with lexicalised grammars, as illustrated by the generation algorithm proposed in [Gardent and Kow, 2006] and the algorithm we will discuss in Chapter 3.

In *Lexicalised TAG* (LTAG), at least one terminal symbol, namely the **anchor**, must appear at the frontier of all initial or auxiliary tree. That is, the anchor is a leaf node of the elementary tree (for instance, *fort* in the auxiliary tree in Figure 2.6). Elementary trees serve as a complex descriptions (e.g. subcategorization and argument realization information) of the anchor.

Two linguistically relevant formal properties of (L)TAG

Extended domain of locality. In a grammar formalism the domain of locality refers to the domain where specifications about different grammatical aspects can be stated together. These grammatical aspects are constituency, constraints, (e.g. agreement),

[8] Either because it corresponds to exactly one or because the commitment to one structure among several has been made.

syntactic and semantic dependencies, word-order and unifications ([Joshi, 1987]). In context-free grammars (CFG) and CFG-based grammar formalisms, the domain of locality is the one level tree corresponding to a rule. Therefore, for instance, the dependency between a verb and its subject and object arguments cannot be specified within the same rule. In contrast, because TAG elementary trees can have arbitrary depth, these dependencies can be specified within the same tree (i.e. domain). The fact that an elementary tree contains the lexical anchor arguments its known as predicate argument co-occurrence.

Factorise recursion from the domain of dependencies. In contrast to CFG(-based) that writes recursion into the phrase structure rules, TAG defines a finite set of simple sentence elementary structures and uses adjunction to produce more complex constructions. Elementary trees are minimal structures where dependencies (e.g. subcategorization and filler-gap) are specified. That is, dependencies are defined locally and made distant through the adjunction operation.

Feature Structures Based LTAG

In brief, a Feature Structure is a set of attribute-value pairs, where a value may be either atomic or another feature structure. Feature structures can be ordered based on the information they contain (i.e. carrying less information or more information). This ordering is known as *subsumption* ordering. *Unification* is a (partial) operation on feature structures. Informally, that is, if the information contained in two feature structures is consistent (unification is possible) they can be combined into a new one containing the information of the two original ones.

Feature structures are used for expressing constraints. Different grammatical frameworks for natural languages incorporate on top of a CFG skeleton a feature unification system to specify certain constraints relative to some linguistic phenomena (e.g. subject-verb agreement). The TAG extension with a feature base unification framework (FB-TAG) was proposed by Vijay-Shanker and Joshi (1988). The objective was to improve the descriptive capacity of TAG. By adding feature structures to elementary trees, it is possible, for instance, to state constraints between dependent nodes within the same elementary tree.

In a FB-TAG, of which an example is given in Figure 2.9, the tree nodes are decorated with two feature structures (called **top** and **bottom**). The operations of substitution and adjunction are then reformulated in terms of unification of appropriate feature structures, thus allowing the constraints on substitution and adjunction to be modelled by the success or failure of unifications. On substitution, the top of the substitution node is unified with the top of the root node of the tree being

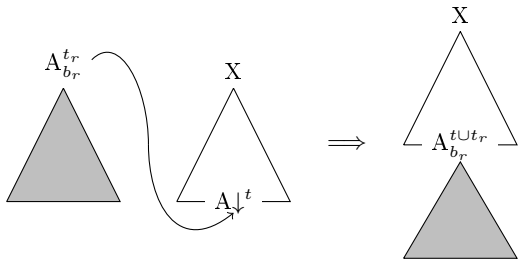


Figure 2.7: Substitution operation in an FB-LTAG

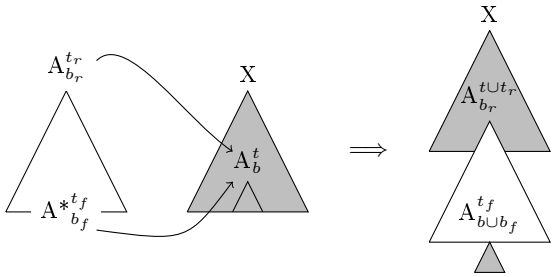


Figure 2.8: Adjunction operation in an FB-LTAG

substituted in. On adjunction, the top of the root of the auxiliary tree is unified with the top of the node where adjunction takes place; and the bottom features of the foot node are unified with the bottom features of this node. Figures 2.7 and 2.8 illustrate this. At the end of a derivation, the top and bottom of all nodes in the derived tree are unified. FB-TAG feature structures are non-recursive and consist of sets of feature/value pairs where the value is either a constant, a disjunction of constants, or a unification variable. Unification variables can furthermore be co-referenced with any other value occurring in the same elementary tree.

Following the example in Figure 2.9, on substitution, the top feature structure of the root node of α_{tatou} will unify on substitution with the NP node of α_{parle} (i.e. $[nb : sg] \sqcup [nb : B]$). On adjunction, the top feature structure of the root NP node of β_{la} will unify with the NP node of α_{parle} (i.e. $[nb : sg] \sqcup [nb : N]$). Furthermore, the initial tree α_{parle} enforces a subject-verb agreement constraint between the verb node and the subject NP node through the feature structures decorating these nodes with the co-referenced variable B.

Besides describing dependencies among nodes within elementary trees, features structures associated with nodes can express other constraints about how trees can combine (or not) with other trees. That is, we can express adjoining constraints (discussed above). Given that adjunction is successful if the unification succeeds, we can state selective constraints (SA) by specifying features values that will cause

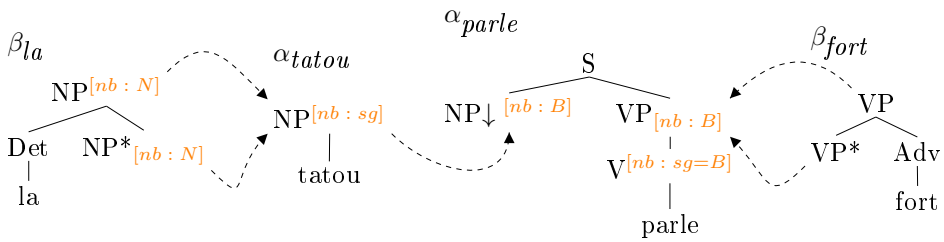


Figure 2.9: Example feature-based tree adjoining grammar (N and B are a unification variables, sg is a constant, and $[f : v]$ is a feature structure with feature f and feature value v).

unification to succeed with desired combinations or fail otherwise. Similarly, given that at the end of the derivation top and bottom features structures are unified, we can model obligatory adjunction (OA) by stating contradicting features in the top and bottom of a node where we want compulsory adjunction.

The small grammar in Figure 2.10 illustrates the implementation of SA with feature structures. The bottom feature structure of the tree $\alpha_{chanter}$ states that the mode of the subordinate clause is equal to infinitive $(md : inf)$. Therefore, the auxiliary tree $\alpha_{veut\text{-}2}$, which takes an infinitive clause as argument, will successfully adjoin into $\alpha_{chanter}$. That is, the bottom feature of the root node in $\alpha_{chanter}$ unifies with the bottom feature of foot node in $\alpha_{veut\text{-}2}$. This is graphically shown by the dotted line arrows from $\alpha_{veut\text{-}2}$ to $\alpha_{chanter}$. From the resulting derived tree we read the sentence *Tammy veut chanter* (Tammy wants to sing). However, the tree $\alpha_{veut\text{-}1}$ will be discarded as unifications during adjunction into $\alpha_{chanter}$ will fail, preventing the generation of the ungrammatical sentence **Tammy veut que chanter* (*Tammy wants that to sing).

2.2.3 TAG derivations

TAG is a tree-generating system: in the derivation process, non-terminals of a tree are rewritten by complete trees. As a result of a derivation a phrase-structure tree is produced which is the derived tree. The set of derived trees constitutes the object language. The structure that records information about the derivation process, i.e. how the elementary trees were combined into a phrase-structure tree is the *derivation tree*. Figure 2.11 shows an example of derived and derivation trees.

In a derivation tree, nodes are labelled with the names of TAG elementary trees. In addition, if the grammar is lexicalised, the nodes would also be named with the lexical items anchoring the elementary trees. Edges are labelled with a description

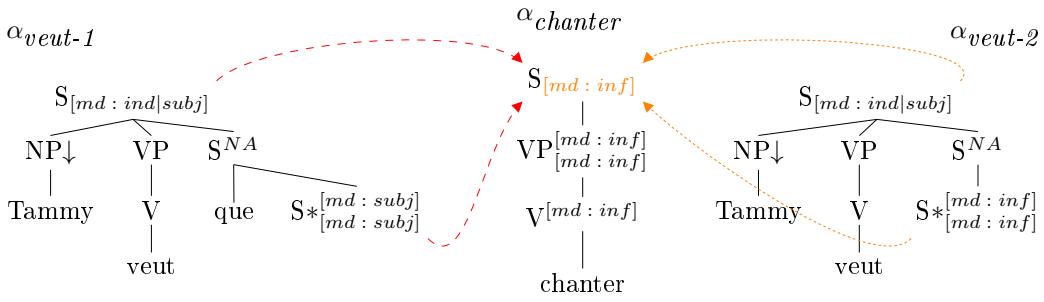
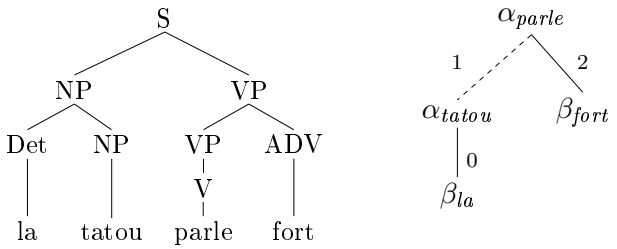


Figure 2.10: Example feature-based tree adjoining grammar illustrating the implementation of SA with feature structures ($[f : v]$ is a feature structure with feature f and feature value v, md is the mode feature, and $inf|ind|subj$ are constant values describing the different mode types).



(a) Derived tree

(b) Derivation tree

Figure 2.11: Parse trees for *La tatou parle fort* (The armadillo speaks loudly) using the grammar of Figure 2.9. In the derivation tree, plain lines indicate adjunction and dotted ones substitution. For simplicity, tree names are replaced with the lemmas anchoring each elementary tree. The number on the upper right of each tree name gives the Gorn address of the node onto which the tree was inserted.

of the operation used to combine the TAG trees whose names label the edge vertices. The edge labels inform the operation type, i.e. substitution or adjunction, and an identification of the node in the target tree where the operation took place, i.e. a Gorn address[9]. When a tree γ is substituted or adjoined into a tree τ, in the derivation tree, the node for γ is said to be a dependent of the node for τ.

LTAG derivation trees are of particular interest due to two key facts: (i) lexicalisation of elementary trees and (ii) encapsulation within elementary trees of the syntactic/semantic arguments of the lexical anchor. In the derivation process, the

[9] "*A Gorn address (Gorn, 1967) is a method of identifying and addressing any interior node within a tree data structure from a phrase structure rule description or parse tree. The Gorn address is a string made up of a series of one or more integers separated by dots, e.g., 0 or 0.0.1. The j-th child of the i-th node has an address i.j.* " (`http://en.wikipedia.org/wiki/Gorn_address`)

substitution and adjunction operations establish links or "dependencies" between lexical items anchoring the composed trees ([Abeillé and Rambow, 2000]). The dependency structures shown by TAG derivation trees have been analyzed in different works. In order to transfer formal properties from TAG to Meaning-Text Theory (MTT, [Melcuk, 1988]), Rambow and Joshi (1994) compare derivation trees with deep syntactic dependency structures, namely DSyntS for MTT. On the other hand, Candito and Kahane (1998) compare derivation trees with semantic dependency graphs, namely SemS in MTT. Although, in both cases, several points of similarities are enumerated, differences between the dependency structures in MTT and derivation trees are reported. Below, we repeat some of the derivation tree features described in these comparisons which underlie the ideas of our approach to semantic-based chart indexing (Section 3.2.2) and syntactic transformations described (Section 4.4.2):

- Function words required by the lexical anchor's subcategorization frame are included in elementary trees and thus not present in the derivation trees (e.g. prepositions).
- Because elementary trees correspond to a semantic unit, nodes in the derivation tree correspond to semantic units (e.g. idioms). As Candito and Kahane (1998) claimed, we can see them as semantic units associated with linguistic information.
- There is one daughter node for each verb argument (due to single substitution steps) while there might be any number of daughter nodes for adjuncts (adjunctions). Though, for predicative auxiliary trees, the direction is inverted: they appear as daughters of the node representing their argument.

2.2.4 FB-LTAG *with semantics*

To associate semantic representations with natural language expressions, the FB-LTAG is modified as proposed by Gardent and Kallmeyer (2003). Each elementary tree is associated with a flat semantic representation. For instance, in Figure 2.12, the trees for *Tex* and *chanter* are associated with the semantics $l_6 : tex(x_6)$ and $l_7 : chanter(e_7, x_7)$, respectively. Importantly, the arguments of a semantic functor are represented by unification variables which occur both in the semantic representation of this functor and on some nodes of the associated syntactic tree. For instance, in Figure 2.12, the semantic index x_7 occurring in the semantic representation of *chanter* also occurs on the subject substitution node of the associated elementary tree. The value of semantic arguments is determined by the unifications resulting

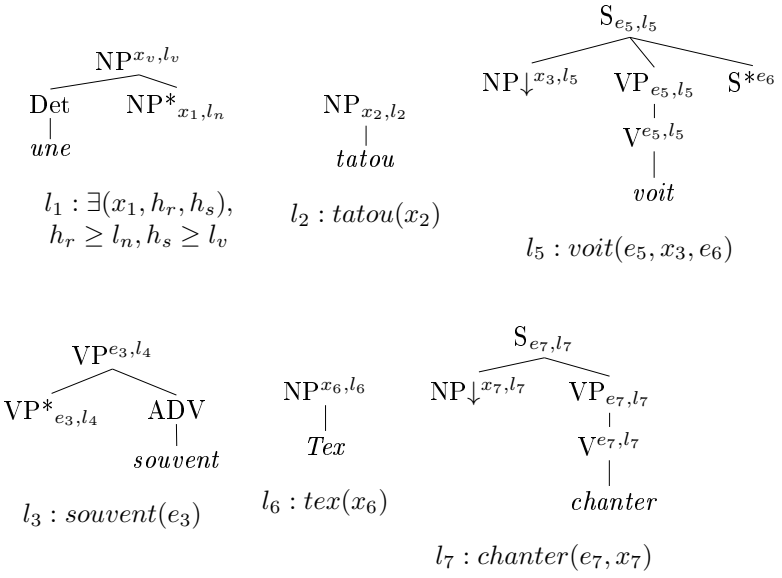


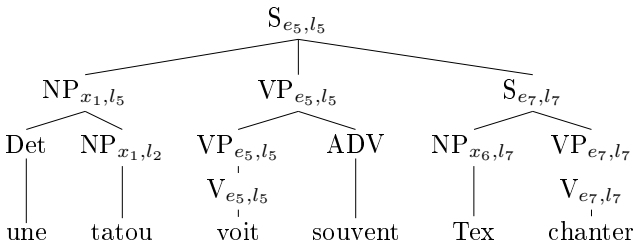
Figure 2.12: An FB-TAG augmented with an unification-based compositional semantics. For the sake of clarity, feature structures are abbreviated, feature percolation has been simplified precluding the possibility that adjunction modifies feature values and only the semantic feature values relevant for semantic construction are indicated. $C^{x,l}/C_{x,l}$ abbreviate a node with category C and a top/bottom feature structure including the feature-value pairs { **index :** x, **label :** l}.

from adjunction and substitution. For instance, the semantic index x_7 in the tree for *chanter* is unified during substitution with the semantic index labelling the root node of the tree for *Tex*. As a result, the semantics of *Tex chante* is $\{l_6 : tex(x_6), l_7 : chanter(e_7, x_6)\}$.

As explained in [Gardent and Kallmeyer, 2003], the semantic representation language used (L_U –underspecified logic) (based on the formalisms defined in [Copestake *et al.*, 2001] and [Bos, 1995]) is a unification-based language which describes first order formulae in the sense that the model of a given L_U formula is a set of first order formulae. For instance, the formula in Figure 2.13 describes the first order formula

$$\exists x_1.(tatou(x_1) \wedge souvent(e_5) \wedge voit(e_5, x_1, e_7) \wedge tex(x_6) \wedge chanter(e_7, x_6))$$

More generally, L_U formulae are flat, underspecified FOL formulae. They are flat in that the tree structure of a FOL formulae is transformed into a conjunction of labelled formulae whereby the label of each formula is used to indicate its position in the initial tree structure. L_U formulae are furthermore underspecified in that the scope of scope bearing operators (quantifiers, modal, negation) is specified by under-



$$l_1 : \exists(x_1, h_r, h_s), h_r \geq l_2, h_s \geq l_5, l_2 : tatou(x_1),$$
$$l_3 : souvent(e_5), l_5 : voit(e_5, x_1, e_7), l_6 : tex(x_6), l_7 : chanter(e_7, x_6)$$

Figure 2.13: Derived tree and semantics for *Une tatou voit souvent Tex chanter* (An armadillo often sees Tex sing).

constrained scoping constraints between so-called holes (written h, h_i) and labels (written l, l_i). Thus, the formulae of L_U consist of labelled elementary predications ($l : R^n(i_1, \ldots, i_n)$ with R an n-ary relation and i_j variables over individuals and/or labels/hole constants), scoping constraints ($h \geq l$ with h a hole constant and l a label constant) and conjunctions (ϕ, ψ with ψ, ϕ formulae of L_U). The models described by L_U formulae are defined by the set of possible "pluggings" i.e., injections from the holes of a formula to the labels of this formula. The following example illustrates this. Suppose the sentence in (14) is assigned the L_U formula (15).

(14) Every dog chases a cat

(15) $l_0 : \forall(x, h_1, h_2), h_1 \geq l_1, l_1 : D(x), h_2 \geq l_2, l_2 : Ch(x, y), l_3 : \exists(x, h_3, h_4), h_3 \geq l_4, l_4 : C(y), h_4 \geq l_2$

Only two pluggings are possible for this formula in (15) namely $\{h_1 \rightarrow l_1, h_2 \rightarrow l_3, h_3 \rightarrow l_4, h_4 \rightarrow l_2\}$ and $\{h_1 \rightarrow l_1, h_2 \rightarrow l_2, h_3 \rightarrow l_4, h_4 \rightarrow l_0\}$. They yield the following (16) meaning representations for (14):

(16) a. $l_0 : \forall(x, l_1, l_3), l_1 : D(x), l_2 : Ch(x, y), l_3 : \exists(x, l_4, l_2), l_4 : C(y)$

b. $l_0 : \forall(x, l_1, l_2), l_1 : D(x), l_2 : Ch(x, y), l_3 : \exists(x, l_4, l_0), l_4 : C(y)$

For more details on the interpretation of L_U and on the semantic representations it permits associating with a grammar of natural language, see [Gardent and Kallmeyer, 2003].

The example in Figure 2.12 shows how a semantic representation is constructed, i.e. how a semantic formula is build for a natural language expression (parsing). However, the FB-TAG equipped with a unification-based compositional semantics

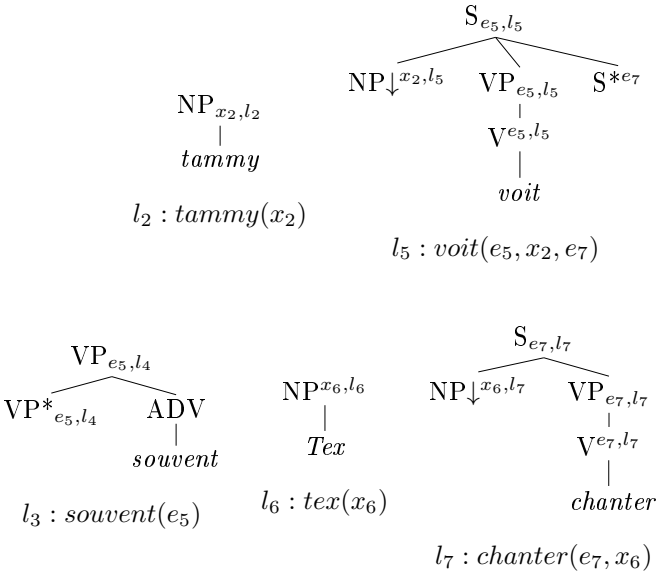


Figure 2.14: An FB-LTAG augmented with a unification-based compositional semantics that produces the sentence *Tammy voit souvent Tex chanter* (Tammy often sees Tex sing) from the given semantic representation $\exists x_2.(tammy(x_2) \wedge souvent(e_5) \wedge voit(e_5, x_2, e_7) \wedge tex(x_6) \wedge chanter(e_7, x_6))$.

is a reversible grammar and thus it is possible to go the other way around, i.e. build a natural language expression for a semantic formula (generation). In the latter case, the semantic formula is given. Therefore, as shown in Figure 2.14, the arguments of the semantic functor $l7 : chante(e_7, x_6)$ are constants values and so are the semantic indices labeling the nodes of its associated syntactic tree (idem for $l_5 : voit(e_5, x_2, e_7)$). During tree combination, the constant values of the semantic indices labeling tree nodes either permit or avoid certain combination of trees. For instance, the substitution of the tree for *Tammy* into the NP node of *chanter* will fail due to the **index** feature-values that do not unify. Preventing in this way the generation of the sentence *Tex voit souvent Tammy chanter* (Tex often sees Tammy sing). Therefore, as a result of tree combination the correct sentence –according to the given meaning representation– *Tammy voit souvent Tex chanter* (Tammy often sees Tex sing) is obtained.

2.2.5 SemTAG

The specific FB-LTAG equipped with unification-based compositional semantics that we use, i.e. SemTAG, is not written tree by tree but instead is compiled from a metagrammar specification. More precisely, the grammar is written and compiled

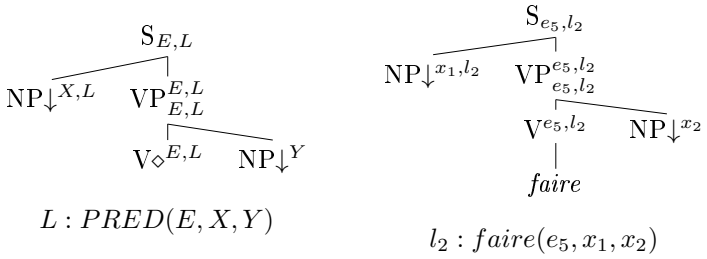


Figure 2.15: Elementary tree schema for a transitive verb (left) and the tree schema anchored by the lemma *faire* (bake) (right).

using the XMG grammar development framework ([Crabbé *et al.*, 2012]).

In a lexicalised TAG, the lexicon consists of a set of elementary structures associated with lexical items (i.e. there is no distinction made between lexicon and grammar). However, concrete TAG implementations (following the approach in XTAG [XTAG Research Group, 2001]) represent this in a factorised way. That is, for instance, instead of repeating exactly the same tree for each verb in the lexicon: *manger* (eat) and *préparer* (to do), they factor out the lexical item and leave the tree as a *tree schema*. The place (node) where the lexical item used to be is called *anchor node* (marked with a $\diamond$), and each time that a tree schema is instantiated with a given lexical item, it is said that the tree schema is *anchored* by the lexical item. Figure 2.15 shows an example of tree plus semantic schema.

SemTAG is implemented following this representation and consists of (i) a set of tree schemas, (ii) a syntactic lexicon associating lexical items with tree schemas (a set of tree schemas grouped into a tree family –as defined in the next paragraph), and (iii) a morphological lexicon associating lemmas with words.

In TAG, an elementary tree is the maximal syntactic projection of a lexical item, i.e. a maximal linguistic representation reflecting the subcategorisation properties of the lexical item. Furthermore, lexical items, for instance, verbs, need to be associated to different elementary trees according to their different voice alternations (e.g. active and passive), the different realisation of their arguments (e.g. cleft and clitic), and the different type of clause they can anchor (e.g. relative clause and interrogative clause). Therefore, in TAG, there might be several elementary trees associated with a lexical item with a given subcategorization frame. All these elementary trees sharing the same subcategorisation frame are grouped together into so-called *tree families*. Figure 2.16 shows some of the trees contained in the family of trees for transitive verbs, e.g. *manger* (eat).

Besides the redundancy eliminated by the factorization into tree schemas and

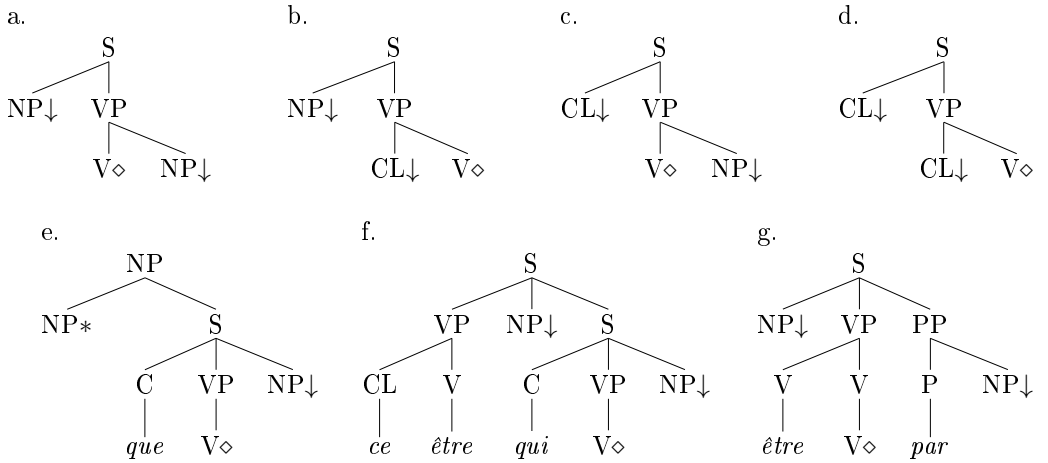


Figure 2.16: Some tree schemas within the transitive verb family. (Note: feature structures and semantics are not shown for the sake of clarity).

syntactic lexicon, XMG permits to take advantage of the fact that there might be several tree schemas per family, and in general in the whole grammar, which share common sub-trees (or *tree fragments*). The idea is that in an XMG metagrammar these shared structures are described only once and then combined to produce complete tree structures. The tree schemas in SemTAG are compiled from an XMG factorised metagrammar.

Succinctly[10], in XMG, the (meta)grammar designer defines (minimal) linguistic descriptions, e.g. phrase-structure tree fragments, which are given a name. Next, these named tree descriptions can be combined by conjunction or disjunction. Some of the trees listed in Figure 2.16 share subtrees corresponding, for example, to a nominal subject noun phrase or share the verb phrase structure. Therefore, these shared tree fragments are defined as a linguistic unit, i.e. a *tree description*, and given a linguistically meaningful name. This name permits referring to the tree description as a whole, then we say that the tree description is *abstracted* by that name. Figure 2.17 shows the tree descriptions used by the XMG rule:

$$ActiveTransitiveVerb \rightarrow CanonicalSubject \wedge Active \wedge CanonicalObject \qquad (2.1)$$

This rule produces the tree (a.) in Figure 2.16. However, it is possible to define

[10]For a complete description of grammar development in XMG we refer the reader to [Crabbé *et al.*, 2012].

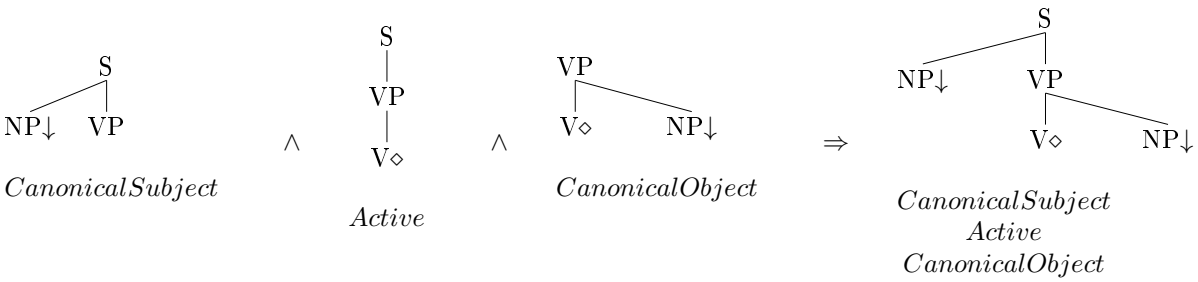


Figure 2.17: Simplified XMG metagrammar example.

more abstract rules. (2.2) below describes the trees (a.), (c.) and (g.) (and the clitic-passive one not shown) in Figure 2.16.

$$Subject \rightarrow (CanonicalSubject \vee CliticSubject) \tag{2.2}$$

$$ActiveTransitiveVerb \rightarrow Subject \wedge Active \wedge CanonicalObject \tag{2.3}$$

$$PassiveTransitiveVerb \rightarrow Subject \wedge Passive \wedge CanonicalObject \tag{2.4}$$

As can be seen in Figure 2.17, the compiled tree is associated with the names of the tree descriptions from which it was built. These tree description names are called *tree properties*. As these tree properties summarise (or abstract) syntactic tree descriptions, it is not needed to analyze the phrase-structure of a given tree γ to know whether it contains a cleft or clitic subject. Instead, the set $P(\gamma)$ of tree properties associated to γ can be used to check whether the tree property *CleftSubject* or *CliticSubject* is a member of the set. We will see in Chapter 4 how tree properties associated to elementary trees are used (i) to specify syntactic constraints over sentences and (ii) to describe syntactically related sentences to generate grammar exercises for language learning.

The two grammar implementations we work with are:

SemFraG. An FB-LTAG for French implemented by Crabbé (2005) and extend with semantics by Gardent (2008).

SemXTAG. An FB-LTAG for English which is a reconstruction of the XTAG grammar ([XTAG Research Group, 2001]) implemented by Alahverdzhieva (2008) and extended with unification-based compositional semantics as described in [Gardent and Kallmeyer, 2003].

2.3 Computer Assisted Language Learning

CALL

Computer-Assisted Language Learning (CALL) is the field concerned with the study, development and use of computer applications in language teaching and language learning ([Levy, 1997; Levy and Hubbard, 2005]). CALL is an interdisciplinary field. Levy's (1997) work reports on its relation to applied linguistics, computer science and psychology. Later on, Kern (2006) and Amaral (2011) stress the importance of CALL research taking into account CALL practitioners (teachers and learners), Second Language Acquisition (SLA) researchers, linguists and computer scientists. In particular, Zock (1996), Nerbonne (2003), and Meurers (2012) highlight the opportunities for and challenges of integrating technology from Natural Language Processing (NLP) research in CALL applications.

interdisciplinary field

tutor, tool, medium view

According to Kern (2006) (in turn extending from Levy (1997)), CALL technology can be seeing to play different roles. Providing (i) instruction, evaluation and feedback in different language areas and skills (e.g. grammar, vocabulary, writing, etc) as a *tutor*, (ii) access to visual, audio or any material useful for language learning, as well as resources and tools such as dictionaries, grammar and style checkers, and concordances for corpus analysis as a *tool*, and (iii) the technological means within which language learning can take place, for instance, sites for interpersonal communication (e.g. multi-user virtual environments), as a *medium*.

activity types

Learning activities in CALL vary from workbook-style exercises, such as Fill-in-the-blank, multiple-choice and matching, to more interactive activities that promote contextualised, language-in-use and culturally informed language learning. For instance, Li and Topolewski (2002) and Amoia *et al.* (2012) embed grammar exercises within a game in a virtual environment, the Croquelandia adventure game ([Sykes *et al.*, 2008]) focuses on pragmatics by teaching how to make request and to apologise in Spanish, and Zip&Terry ([Li and Topolewski, 2002]) and Thethis ([Segond *et al.*, 2005]) provide dialogues in simulated situations. Interactive or situated learning activities provide settings that favor "unconscious" processes involved in second language acquisition ([Krashen, 1982]). Nevertheless, learner "awareness" of linguistic phenomena in the target language has been shown to be important to foster learning ([Long, 1991; Schmidt, 1995; Long, 1996]).

Not all of the existing CALL applications make use of NLP techniques. For instance, *Hot Potatoes* (`http://hotpot.uvic.ca/`) provides a set of tools which facilitate the manual edition of exercises: the exercise question, the expected solution and adequate feedback. However, these tasks can be automated. In the search for automation, Intelligent CALL (ICALL) applications incorporate techniques from

ICALL

NLP and expert systems. Current NLP research for language learning focuses on providing support for (i) the creation of learning material and (ii) the detection and evaluation of learner errors and generation of personalised feedback.

activity types and learner input

Most of the learning material created by ICALL applications consists of so-called *objective test items*. That is, test items such as multiple-choice questions, Fill-in-the-blank and cloze exercise items, whose answer is strongly constrained (the range of possible answers is very small or fixed) and can therefore be predicted and checked with high accuracy. There also exist less constrained activity types, such as Fill-in-the-blank without hints other than the sentence wherein the blank appears or make-a-sentence exercises from a given set of words, which expect a larger set of correct answers. Bachman and Palmer (1996) name the first type of activities *selected response* and the latter *limited production*. According to Bachman and Palmer, the relation between a test item of a given activity type (input given to the learner) and the criteria for correctness (the set of expected correct answers) is an important point in the evaluation of usefulness in test development. Moreover, the learner's input is, in general, evaluated through string matching against a target correct answer (or a set thereof) and the type of feedback is restricted to "correct/incorrect". Automatically providing detailed feedback is a difficult task which involves interpreting the, often ill-formed, learner input, determining errors and proposing a correction.

objective test items

learner input analysis

The motivations for designing activities that constrain the learner input are discussed in [Amaral and Meurers, 2011]. First, parsers cannot rely on the lexical and syntactic properties of the language to reduce the search space. Because these regularities might not be followed in the learner language. In addition, techniques such as *mal*-rules (augmenting the grammar with rules that describe learner errors) or relaxation (eliminate certain constraints from the grammar) increases the parsing complexity. Second, a possibly more difficult task than the evaluation of form is *meaning evaluation* (e.g. determining whether the answer given by the learner to a reading comprehension question is correct). ICALL systems generally evaluate meaning based on surface form; therefore, as there are many ways in human languages to express the same meaning ICALL systems should be able to deal with significant variation in learner input.

feedback generation

To provide detailed and personalised feedback, an ICALL system needs not only to model the target language and be able to analyze learner input. It might also need to represent the learner knowledge about the target language as well as the instructional scheme that the system might enforce; i.e. a *learner model* and an *instructional model*. For instance, Michaud and McCoy (2000) propose a multi-component model where the representation of the learner grammar proficiency is

used both to evaluate learner input and to automatically deliver instruction that is, as they said, "at the frontier" of learner competence (cf. [Krashen, 1982; Vygotsky, 1986]).

Our NLG-based approach aims at the creation of grammar textbook style of exercises items. It can be thought as a component part of a language tutoring system. That is, it could be embedded into a system that implements a learner model, evaluates learner input and assesses learner language skills, provides detailed feedback and proposes an instructional model. Furthermore, our exercise generation approach can be integrated with 3D environments resulting in more interactive activity ([Amoia *et al.*, 2012]). On one hand, it is more meaningful as the content of the exercise goes with the virtual environment. On the other hand, it might be more engaging as it is combined with gaming elements. We show how the NLG-based approach supports the creation of exercises with a selected answer (e.g. Fill-in-the-blank giving enough information so as to reduce the possible answers to one) as well as limited production exercise types (e.g. Shuffle or rewrite-a-sentence type of exercises where there might be different correct alternative answers). In the next section, we review related work on the generation of learning content and learning activities.

2.3.1 Automatic authoring of learning material

As aforementioned, much of research works in ICALL address the creation of learning material relying on NLP techniques. Two major subtasks are in focus. One is concerned with the search of appropriate text material and the other one with the creation of learning activities and test items.

Text retrieval

Some research work in CALL has focused on the *readability-based* retrieval of text. The main goal is to develop CALL tools that facilitate reading and vocabulary practice. This work aims at selecting text that is appropriate to the learner comprehension capabilities (i.e. comprehensible learner input [Krashen, 1982]) and in correspondence with a given learning stage ([Pienemann, 1998]). To this end, most of the proposed approaches rely on readability measures computed on documents selected from large corpora or the web.

Heilman *et al.* (2008) describe a search system, namely REAP Search. A customised search interface permits the specification of pedagogical constraints: reading level and text length, in addition to topic and a set of target words. This interface accesses a database of documents that is created by crawling the web based on a

set of target words and further annotating and filtering the retrieved documents by different criteria. Some of the used criteria are date of the document, size, reading level and text quality. They use a language model that predicts reading level. While in the current approach predictions are based on lexicon models, a new approach is being explored which also includes syntactic features. The objective behind this is to extend the available criteria for selecting text allowing the teachers to focus on grammatical features, e.g. choose a text with simple grammar constructions.

The LAWSE prototype ([Ott *et al.*, 2010]) develops a specialised search engine which allows teachers to look for texts that satisfy content and relevant language properties requirements. This approach associates to each document a set of summary properties (i.e. a set of property-value pairs). These properties could summarise any relevant feature which could be associated to the text and that admits a unique value. For instance, the number of words in the text, the ratio of gerunds to all verb forms or, any readability measure. In addition to the regular query terms specified by users, queries contain a set of constraints based on these properties which specify the desire range of values for each property.

Heilman *et al.* (2008) show that a great deal of document filtering is needed (1% of the originally downloaded documents remain in the database of selected documents) as well as the final teacher judgment on the select text. While they are valuable tools providing teachers and intermediate or advanced learners with "real life" texts for reading and vocabulary training, it is not clear how much they would help in producing learning material for beginners or that targets training of specific grammar points. Both projects agree that the web might not provide enough reading material for beginners. As a solution to this issue, Ott *et al.* (2010) propose the use of text simplification techniques.

Automatic generation of exercise and test items

Developing exercise and test items manually is a time-consuming task. Mitkov *et al.* (2006) evaluate the efficiency gain in using a computer-aided environment to generate test items and the quality of the generated items. The efficiency is measured by comparing against manual edition of test items, they found that when using the system the teacher spends in average 1 minute and 36 seconds against 6 minutes and 55 seconds in average per item by doing it manually[11]. To evaluate the quality of the items, they run two experiments in which students were tested using a set of both (semi-)automatically and manually generated test items. They also found that

[11] This numbers are related to the "first post-editing experiment" (see [Mitkov *et al.*, 2006]), similar results are reported for a second one.

the quality of the (semi-)automatically generated items is acceptable and compares favourably with hat of manually produced exercices.

Thus, an important strand of research in ICALL addresses the automation of exercise specifications relying on NLP techniques ([Mitkov *et al.*, 2006; Heilman and Eskenazi, 2007; Karamanis *et al.*, 2006; Chao-Lin *et al.*, 2005; Coniam, 1997; Sumita *et al.*, 2005; Simon *et al.*, 2010; Lin *et al.*, 2007; Lee and Seneff, 2007]). Mostly, this work targets the automatic generation of objective test items. These approaches use large corpora and machine learning techniques to automatically obtain the *stems* (sentences from which the exercise is built), *questions* (exercise question), the *keys* (correct answers) and the *distractors* (incorrect answers) that are required by such test items. Example (17) shows a multiple-choice exercise item illustrating this terminology, i.e. the parts we identified in the process of generating an exercise item. [12]

(17) (Q): *Do you have a dictionary? Yes, I have a good .....*
a. some b. one c. another d. it e. mine

stem: *Do you have a dictionary? Yes, I have a good one.*
question: *Do you have a dictionary? Yes, I have a good .....*
keys: *one*
distractors: *some, another, it, mine*

Mitkov *et al.* (2006) generate multiple-choice questions to test factual knowledge conveyed in a given source text. To do this, the question generator identifies a term in the given text, then, by using simple rules transforms the sentence containing the term into a *wh*-question sentence, finally, it chooses a set of distractors. To extract terms, they identify nouns and NPs after parsing the input text, and then use various criteria (e.g. most frequent term and NP following certain regular expression patterns) to select the candidate terms. An example of test item from [Mitkov *et al.*, 2006] is shown in (18).

(18)
stem (source clause): *Transitive verbs require objects.*
question: *Which kind of verbs require objects?*
keys: *transitive verbs*
distractors: *modal verbs, phrasal verbs, active verbs*

Heilman and Eskenazi (2007) generate "related words" questions using an automatically extracted thesaurus. Their generated items target vocabulary practice and assessment, given a word the learner should point to the appropriate set of related

[12] Example extracted from `http://a4esl.org/`.

words (e.g. words that are near match or opposites). An example of generated item from [Heilman and Eskenazi, 2007] is shown in (19).

(19)

question: *Which set of words are most related in meaning to "reject"?*
keys: *{accept, oppose, approve}*
distractors: *{pray, forget, remember}, {invest, total, owe}, {persuad, convince, anger }*

Simon *et al.* (2010) generate multiple-choice questions that aim at assessing vocabulary in context. They rely on a distributional thesaurus and collocations to build the items. Given a key word, they produce an item where the distractors are the set of words related to the key word, according to a thesaurus, but which do not occur with a key's collocation. Then, a sentence is retrieved from a corpus that contains the key word and the collocation. Chao-Lin *et al.* (2005) also target the generation of multiple-choice items targeting the evaluation of vocabulary in context. That is, given a word and its intended meaning, they use word sense disambiguation techniques to find sentences in which the word is uses with the intended meaning.

Lin *et al.* (2007) design multiple-choice questions that aim at understanding the meaning of adjectives in a given text. Lee and Seneff (2007) generate multiple-choice items focusing on prepositions.

All these approaches to the computer-aided generation of test (and exercise) items have shown to be useful in reducing the time spent by language instructors while still providing items of quality comparable to those manually produced. Nevertheless, these approaches focus mainly on reading comprehension and vocabulary, perhaps with the exception of Lee and Seneff (2007) who focus on prepositions. Further, the type of items are mostly intended for testing intermediate or advanced learners.

There is some work, however, that targets the generation of *grammar exercises*. By grammar exercises, we mean those exercises that aim at practicing specific linguistic phenomena of the targeted language, for instance, prepositions, pronouns, verb voice, verb forms, etc. Thus, Chen *et al.* (2006) describe a system called FAST which supports the semi-automatic generation of multiple-choice and error detection exercises while Aldabe *et al.* (2006) present the ArikiTurri automatic question generator for constructing Fill-in-the-blank, Word Formation, multiple-choice and error detection exercises. These approaches are similar to the approach we propose in Chapter 4. First, a bank of sentences is built which are automatically annotated with syntactic and morpho-syntactic information. Second, sentences are retrieved from this bank based on their annotation and on the linguistic phenomena the exercise is meant to illustrate. Third, the exercise question is constructed from the

grammar exercises

retrieved sentences. There are important differences however.

First, in these approaches, the source sentences used for building the test items are selected from corpora. As a result, they can be very complex and most of the generated test items are targeted for intermediate or advanced learners. In addition, some of the linguistic phenomena included in the language schools curricula may be absent or insufficiently present in the source corpus ([Aldabe *et al.*, 2006]). In contrast, our generation based approach permits controlling both the syntax and the lexicon of the generated exercises.

Second, while, in these approaches, the syntactic and morpho-syntactic annotations associated with the bank sentences are obtained using part-of-speech tagging and chunking, in our approach, these are obtained by a grammar-based generation process. As we shall see in Chapter 4, the information thus associated with sentences is richer than that obtained by chunking. In particular, it contains detailed linguistic information about the syntactic constructs (e.g., cleft subject) contained in the sentences in the generation bank. This permits a larger coverage of the linguistic phenomena that can be handled. For instance, we can retrieve sentences which contain a relativised cleft object (e.g., *This is the man whom Mary likes who sleeps*) by simply stipulating that the retrieved sentences must be associated with the information *Cleft Object*).

To sum up, our approach differs from most existing work in that it targets the production of syntactically and lexically controlled grammar exercises rather than producing grammar exercises based on sentences extracted from an existing corpus.

2.3.2 Natural Language Generation in CALL

As we suggested in Chapter 1, NLG techniques have been little explored within the context of ICALL. Here, we discuss four NLG-based approaches. They are different to ours in terms of goals and generation frameworks though in some cases, ideas overlap.

One of the systems is a writing tutor called COMPASS-II ([Harbusch *et al.*, 2008b; Harbusch *et al.*, 2009; Harbusch and Kempen, 2010]). COMPASS-II provides a writing environment for second language learning of German which markedly differs from others. Instead of letting the learner produce a text and then parsing it to check grammaticality and provide feedback, their approach uses a generator to assist the learner in an incremental ("scaffolded") sentence production providing in this way accurate feedback. The system relies on a grammar-based (Performance Grammar (PG, [Kempen and Harbusch, 2002])) generator. Grammatical information is encoded in lexical entries, called elementary treelets. In its graphical interface, the

learner can select words from a lexicon, anchoring a treelet, and compose them into a sentence (or phrase). The system (i) monitors the structure (i.e. treelet) combination and (ii) verifies, on demand by the learner, the final word order assigned to the sentence. In both cases, it provides feedback based on the information provided by the generator. What COMPASS-II and our approach have in common is that both rely on a same concept: *paraphrases*. In their case, it is exploited to (a.) evaluate the final phrase tree structure created by the learner comparing against all possible structures licensed by the grammar for the chosen lexical items and (b.) on demand show to the learner of all possible realisations for the chosen set of words. In our case, it is exploited to generate controlled and varied grammar exercises.

The "Sentence Fairy" ([Harbusch *et al.*, 2007; Harbusch *et al.*, 2008a]), based on the same paraphrase generator, is a tutor system to teach essay writing to elementary-school children (indeed German as first language). The tutor simulates a "writing conference"[13] scenario. Three exercise types are available: story reconstruction, sentence combining, and word order. Exercise items of these type are automatically generated based on a story abstract representation. The system provides an interface, the "teacher mode", where the instructor can enter simple clauses making up a story and specifies rethorical relations thereof ([Harbusch *et al.*, 2012]). The clauses are parsed (using the same grammatical framework) and stored in a kind of MRS representation (cf. Section 2.1). Rethorical relations between clauses are also stored as predications. For instance, when producing a sentence combining exercise, the generator will produce a set of possible syntactic realisations for each of two sentences together with verbalisations of the rethorical relations. The task of the learner is to choose an adequate combination of them. The Sentence Fairy system exploits sentence generation in a similar manner we do. The underlying idea being to generate varied exercises from few simple input (this will be discussed in 4.3.1).

Both COMPASS-II and Sentence Fairy systems show how NLG can be used to provide limited production type of activities getting around the issues of evaluating ill-formed input from the learner.

Focused on error detection and providing relevant feedback, Zamorano-Mansilla (2004) describes an approach based on the KPLM sentence generator. Concretely, he focused on detecting errors in Fill-in-the-blank exercises and providing relevant feedback. The exercise items (i.e. question and correct answer) are build automatically from stored sentences. To evaluate whether the word provided for the blank

[13] The text of a story is read and understood, and modified to get a better coherent and fluent version discussing alternative syntactic constructions and lexical choices in groups under the direction of a tutor (a teacher).

by the learner is correct, the system compute the paths through systemic-grammar (SFG, [Halliday, 1985]) for both solutions (the expected answer and the answer provided by the learner) and compiles a set of grammatical features for each. Feedback on the origins of learner errors is given based on these sets of features, explaining the difference between the learner and correct solution. For instance, in Zamorano-Mansilla's example, for the given exercise item *Don't put your feet ..... the table*, if the learner answer is the preposition *with*, the system will provide feedback using the set of features collected for her answer, i.e. {`accompaniment-process`}, and those for the correct answer, i.e. {`spatio-temporal-process`}.

Zock and Quint (2004) focus on the generation of exercises based on templates and entries from a dictionary. Their approach is goal-driven as each syntactic template is associated with a goal (e.g. comparison or definition) and the learner chooses the sentence to be generated based on intention (goal) rather than on the syntactic form (template) itself. Our exercise generation approach, by relying on a grammar-based generator, focuses on providing varied and syntax-controlled exercise types. It would be interesting to integrate in our grammar a pragmatic or functional dimension as proposed in [Zock and Quint, 2004].

In this chapter, we have situated our generation approach and highlighted the complexity issues that we will address in Chapter 3. We have described in detail the grammatical framework, namely *SemTAG*, in which our generator builds upon. We will make use of the definitions, properties and features discussed in this chapter about *SemTAG* in the remaining of the thesis. In Chapter 4, we will put to work the *SemTAG*-based generation approach for the generation of grammar exercises for language learning.

Chapter 3

Optimising surface realisation

Contents

Each sentence derivation in a TAG ([Joshi and Schabes, 1997]) yields both a derived tree representing the phrase structure of the sentence and a derivation tree specifying how the elementary TAG trees used to build this derived tree were combined (cf. Section 2.2). Interestingly, the derivation trees generated by TAG form a regular tree language ([Vijay-Shanker *et al.*, 1987]). Furthermore, TAG derivation trees have been shown to provide an intermediate representation from which both a sentence and its semantic representation can be derived ([De Groote, 2002;

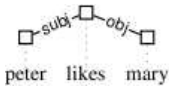
Pogodalla, 2004; Kanazawa, 2007; Shieber, 2006]). They also have been shown to facilitate the reformulation of sentence generation with TAG into a constraint problem ([Koller and Striegnitz, 2002]) or a planning problem ([Koller and Stone, 2007]) for which optimisation techniques exists. In other words, TAG derivation trees provide a pivot language which supports both parsing (going from a sentence to its possible syntactic structures and semantic representations) and generation (going from a semantic representation to one or more sentences). Moreover, derivation trees constitute a simpler language keeping the necessary information from the derived language (e.g. how elementary trees are combined) and leaving aside details about phrase structure.

In this chapter, we propose a sentence realisation approach for FB-TAG equipped with a compositional unification-based semantics which makes use of a feature-based Regular Tree Grammar of TAG derivation trees (FB- RTG, [Schmitz and Le Roux, 2008]) translation. We show how the use of this FB-RTG based approach, while providing an exact grammar of TAG derivation trees (the FB-TAG-toFB-RTG translation preserves all feature information) permits optimising surface realisation.

3.1 Introduction

The high worst-case complexity of surface realisation from flat semantics stems from the interaction between the lack of ordering constraints in the input and lexical ambiguity (cf. Section 2.1). Contrary to parsing where the input is a string, i.e. an ordered list of words, the input to surface realisation is a bag of literals. This lack of constraints on the input potentially results in an unguided exploration of all possible combinations which in effect is exponential in the number of literals present in the input semantics. Moreover, in a lexicalist grammar, such as *SemTAG*, a given literal is associated with many different grammatical structures, the number of possibilities to be explored is very high.

Various techniques have been proposed to cope with this combinatorics and help improving practical run-times. To restrict the combinations tried during generation, Kay (1996) and Carroll and Oepen (2005) propose a chart-based generation algorithm in which only constituents with non overlapping semantics and compatible indices are considered for combination. Kay (1996), Carroll *et al.* (1999) and Gardent and Kow (2006) propose various techniques to restrict the combinatorics induced by intersective modifiers all applying to the same structure. To lessen the effect of lexical ambiguity, Koller and Striegnitz (2002) and Gardent and Kow (2007) describe two alternative techniques for reducing the initial search space. Carroll and Oepen



word	labels	valency
likes	∅	{ subj,obj,adv* }
Peter	{ subj,obj }	∅
Mary	{ subj,obj }	∅

Figure 3.1: Example of TDG parse tree and lexicon.

(2005) use "packing" to reduce the number of structures built during generation by grouping equivalent ones. We will discuss related work on efficient surface realisation in Section 3.4. However, first, we present Koller and Striegnitz's (2002) proposal from which ours is inspired.

3.1.1 On the generation of derivation trees

Koller and Striegnitz's (2002) proposal arises from the observation that surface realisation from flat semantics has combinatorial properties similar to the problem of parsing free word order languages. A natural question then is whether it is possible to make use of existing tools for parsing free word order languages for generation from flat semantics. They propose to encode generation with TAG as dependency parsing with Topological Dependency Grammar (TDG, [Duchier and Debusmann, 2001]), which together with its parser, is developed considering the problem of parsing free word order languages.

In a TDG grammar, the parse tree is an unordered tree whose nodes are in one-to-one correspondence with the words in the sentence and whose edges are labelled with syntactic relations. Word order in TDG is initially free but there is a separate mechanism to specify constraints on linear precedence. The lexicon associates words with a set of lexical entries each of them specifying different dependency constraints. Each lexical entry specifies two sets of labels: *labels* is the set of allowed labels as incoming edges and *valency* the set of allowed outgoing edge labels. See Figure 3.1[14] for an example of TDG parse tree and TDG lexicon corresponding to the sentence *Peter likes Mary*. In this example, the lexical entry for *likes* specifies that it does not accept any incoming edge, thus it could only be used to label a root node in the tree, but requires two compulsory outgoing edges, labelled `subj` and `obj`, and any number of outgoing edges labelled `adv`. The entries for *Peter* and *Mary*, on the contrary, do not allow any outgoing edge but require one incoming edge whose label should be in the labels set: `subj, obj`. The parser builds all trees that can be built using a lexical entry for each word in the input sentence.

[14] Example extracted from [Koller and Striegnitz, 2002]

Koller and Striegnitz's encoding. The TAG variant they consider is a TAG grammar ([Joshi and Schabes, 1997]) without feature structures but with a syntax-semantics interface in which non-terminal nodes of elementary trees are decorated with semantic indices (an example grammar is shown in Figure 3.2). Each elementary tree is associated with a semantic representation.

The idea underlying this encoding is to obtain a dependency grammar that models how a TAG elementary tree could be use in a derivation. That is, given a TAG elementary tree, (i) which operations (substitution or adjunction) can take place at which of its nodes and (ii) how the given tree could be combined (substituted or adjoined) into another tree. The derived dependency grammar consists of a set of lexical entries each of them derived from a TAG elementary tree in the original grammar. These lexical entries are associated with two sets of constraints. One of the sets, called *valency*, models point (i) and the other, called *labels* models point (ii).

The encoding can be summarised as follows. Let us assume a TAG grammar G as introduced in previous paragraphs (for which an example was provided in Figure 3.2). The encoding makes use of two types of edge labels: substitution (`Subst`) and adjunction (`Adj`). An edge with a substitution label $\texttt{Subst}_{A,i,p}$ from α to β indicates that β should be plugged into the p-th substitution node in α that has label A and index i. $\texttt{subst}(A)$ is defined as the maximum number of occurrences of A as the label of substitution nodes within any elementary tree of G, and p takes values from $\texttt{subst}(A)$. An edge with an adjunction label $\texttt{Adj}_{A,i}$ from α to β specifies that β adjoins into some node in α with label A and index i admitting adjunction. Given an elementary tree τ in G with root label category A, root semantic index i, and lexical anchor w, a lexical entry l_w is built in the following way:

- *labels* set
  - if τ is an initial tree, then l_w accepts incoming edges labelled as $\texttt{Subst}_{A,i,p}$ where $p \in \texttt{subst}(A)$.
  - if τ is an auxiliary tree, then l_w accepts incoming edges labelled as $\texttt{Adj}_{A,i}$. Each label $\texttt{Adj}_{A,i}$ encodes an adjunction site which is determined by the category A and index i.
- *valency* set
  - l_w requires one outgoing edge label for each substitution node of τ labelled as $\texttt{Subst}_{B,x,p}$, where B is the category label of the node, x the semantic index decorating the node, and p is the p-th substitution with label $B : x$ in τ.

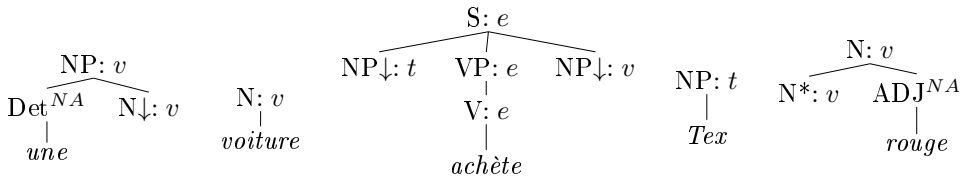


Figure 3.2: An example of TAG grammar variant used in Koller and Striegnitz for the French version of the sentence *Tex achète une voiture rouge* (Tex buys a red car), with semantics { *tex(t), achète(e, t, v), voiture(v), rouge(v)* }.

atom	labels	valency
start	$\emptyset$	{ $\texttt{Subst}_{S,e,1}$ }
achète	{ $\texttt{Subst}_{S,e,1}$ }	{ $\texttt{Subst}_{NP,t,1}$, $\texttt{Subst}_{NP,v,2}$, $\texttt{Adj}_{VP,e}*$, $\texttt{Adj}_{V,e}*$ }
tex	{ $\texttt{Subst}_{NP,t,1}$, $\texttt{Subst}_{NP,t,2}$ }	{ $\texttt{Adj}_{NP,t}*$ }
indef	{ $\texttt{Subst}_{NP,v,1}$, $\texttt{Subst}_{NP,v,2}$ }	{ $\texttt{Adj}_{NP,v}*$, $\texttt{Subst}_{N,v,1}$ }
voiture	{ $\texttt{Subst}_{N,v,1}$ }	{ $\texttt{Adj}_{N,v}*$ }
rouge	{ $\texttt{Adj}_{N,v}$ }	$\emptyset$

Table 3.1: Encoding of the grammar in Figure 3.2

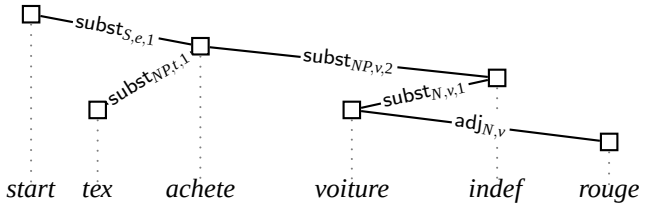


Figure 3.3: Dependency tree

– l_w requires an arbitrary number (possibly zero) of outgoing edge labels for each adjunction site in τ labelled as $\texttt{Adj}_{B,x}$, where B is the category of the node and x the semantic index decorating the node.

A special entry for the *start* symbol is added with valency $\texttt{Subst}_{S,k,1}$, where S is the root category and k is the semantic index with which generation should start. By applying the encoding to the TAG grammar in Figure 3.2 the dependency grammar shown in Table 3.1 is obtained (column atom refers to the semantic predicate).

Parsing with the encoded grammar produces a parse tree (an example is shown in Figure 3.3) which is very close to a TAG derivation tree: nodes are labelled with elementary trees, edges represent substitution and adjunction operations. From this parse tree (derivation tree) the derived tree could be trivially constructed.

Generation using a constraint based dependency parser. Koller and Strieg-

nitz's (2002) approach showed favorable results in its evaluation and comparison with other approaches. They achieved comparable running times to those obtained in [Carroll *et al.*, 1999] (and reported later improvements of Carroll *et al.*'s (1999) algorithm) for the following sample sentences:

(20) *The manager in that office interviewed a new consultant from Germany.*

(21) *Our manager organised an unusual additional weekly departamental conference.*

Earlier on, Gardent and Thater (2001) proposed using a constraint based parsing approach for generation with a variant of TAG: in which elementary tree descriptions are associated with semantics and tree nodes are decorated with semantic indices. They use a Description Grammar (DG) encoding where the basic building units are tree descriptions instead of trees. The DG takes an axiomatic view of the grammar (rather than generative, i.e. derived trees constructed as a sequence of rewriting steps). In their encoding, a tree description is a conjunction of literals that specify either the label of a node (with combining constraints) or the position of a node relative to other nodes (dominance links). The authors report that the performance of the constraint-based generation approach decreases with the length of the input.

Several remarks need to be done about what has been discussed so far. First, both approaches, i.e. Gardent and Thater (2001) and Koller and Striegnitz (2002) approaches, benefit from the constraint-based encoding of the global constraints stated by the grammar. In particular, the propagation step, once a decision (or choice) in the process of generation has been taken, filters out those variable values which are inconsistent with the current state of the problem. In other words, the propagation step dynamically prunes the search space.

Second, a difference between the approach in [Gardent and Thater, 2001] and the approach proposed by Koller and Striegnitz (2002) is that the former builds *derived* trees while the latter builds *derivation* trees. By building the derivation tree, the second approach only needs to take into account essential information about how trees could be combined. In contrast, the first approach needs to keep track of the internal structure of the elementary tree.

Finally, Koller and Striegnitz's (2002) constraint based dependency parser implements two important mechanisms to deal with ambiguity and intersective modifiers. The so-called "selection constraints" restrict which lexical entry should be selected for a node. When more than one lexical entry applies for a node and they contain "shared information" this information is assigned to the node without committing to any thereof. In other words, the parser implements a packing mechanism. Furthermore, they allow for multiple adjunctions in a node, this permits to get rid of

the 2^n intermediate structures as well as the $n!$ possible ways to order the n given modifiers.

It is difficult to assess how much of the efficiency is due to the parser and how much to the grammar conversion. Intuitively however, the motivation underlying the construction of a derivation rather than a derived tree is that efficiency might be increased because the context-free derivation trees (i) are simpler than the mildly context sensitive trees generated by an FB-TAG and (ii) permit drawing on efficient parsing and surface realisation algorithms designed for such grammars.

3.1.2 Using the RTG encoding of TAG derivation trees: RTGen

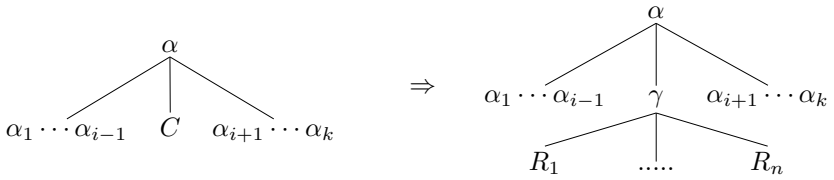
The encoding proposed by Koller and Striegnitz (2002) models how a TAG elementary tree can be used in a derivation leaving phrase structure information aside. The parse tree obtained in Koller and Striegnitz's approach can be seen as a TAG derivation tree. Its nodes are labelled with TAG elementary trees and the edges are labelled with the type of operation and the tree node at which the operation took place. In fact, their dependency grammar encoding of TAG describes the sort of dependency structures shown on TAG derivation trees.

Taking inspiration from Koller and Striegnitz's (2002) approach, we formulate the TAG based surface realisation task as the problem of building derivation trees rather than derived trees. However, to model TAG elementary trees participation in a derivation we rely on another grammar formalism: *feature-based regular tree grammars* ([Schmitz and Le Roux, 2008]).

The language of TAG derivation trees is a regular tree language, i.e. can be generated by a regular tree grammar (cf. [Vijay-Shanker *et al.*, 1987; Shieber, 2006]). Different (equivalent) encodings of regular tree grammars of TAG derivations exist, but alike Koller and Striegnitz (2002) they do not take into account feature structure information. Based on this observation, Schmitz and Le Roux (2008) define feature-based regular tree grammars and a translation from FB-TAG to FB-RTG. The translation of the feature structures allows the transfer of all the linguistic information from the FB-TAG of derived trees to the FB-RTG of derivation trees.

Intuitively, each TAG elementary tree is represented as the fragment it contributes to a derivation tree. Each elementary tree is encoded as a rule of an RTG grammar ([Comon *et al.*, 1997], we given the RTG definition in the next section) of the form $C \rightarrow \gamma(R_1 \cdots R_n)$ where: (i) C defines the type of contribution the elementary tree provides to a derivation, i.e. in which operation it could participate in, (ii) $R_1, \cdots, R_n$ describe the requirements of the elementary tree within a derivation, i.e. which type of operations should/might take place on that tree, and (iii) γ is the

elementary tree name. The diagram below illustrates a step in a derivation where a tree α has been combined with the trees $\alpha_1 \cdots \alpha_{i-1}$, $\alpha_{i+1} \cdots \alpha_k$ and γ. In turn, the tree γ needs to be completed and requires either the adjunction or substitution of TAG elementary trees satisfying $R_1 \cdots R_n$.



This is very close to the lexical entries of the dependency grammar encoding of Koller and Striegnitz (2002), the right-hand side corresponds to the incoming edges or *labels* set and the second to the outgoing edges or *valency* set.

Contrary to the approaches in [Gardent and Thater, 2001] and in [Koller and Striegnitz, 2002], this encoding embodies a generative view of grammar: derivation trees are constructed as a sequence of rewriting steps. To build derivation trees with FB-RTG we need an algorithm that implements the derivation relation defined for RTG, the major requirement here being that the algorithm should be "guided" by the input semantics. To this end, we implement an algorithm that is a variant of the Earley algorithm ([Earley, 1970]) which provides a *mixed* strategy for the construction of the derivation tree: both top-down predictions and bottom-up completions. We developed a surface realiser based on this algorithm called RTGen.

RTGen's generation algorithm also needs to deal with lexical ambiguity and the lack of ordering information in the input flat semantics. RTGen draws on Kay's (1996) and Carroll *et al.*'s (1999) chart generation approaches, and makes use of the now standard semantic criteria proposed in [Kay, 1996; Carroll *et al.*, 1999] to reduce the number of combinations tried out by the realiser. Based on ideas from [Carroll and Oepen, 2005], RTGen implements a local ambiguity packing technique which has shown to improve realisation run-times in practice.

In what follows, we first present the FB-TAG to FB-RTG translation (Section 3.1.3), we then present the RTGen realiser we developed (Section 3.2).

3.1.3 Converting SemTAG to FB-RTG

First, we will present the TAG to RTG conversion i.e., the conversion for a grammar without feature structures. Then, we go on to indicate how feature structures are converted. For a complete description of this FB-TAG to FB-RTG conversion, we refer the reader to [Schmitz and Le Roux, 2008].

A *regular tree grammar* ([Comon *et al.*, 1997]) is a grammar whose rules rewrite a non-terminal symbol as a tree whose internal nodes are each labelled with a terminal symbol and whose leaf nodes are each labelled with a terminal or non-terminal symbol.

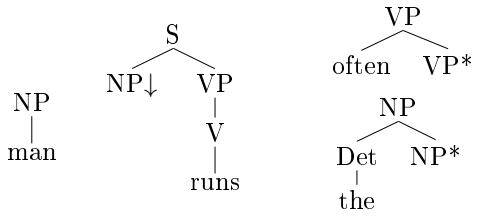
Formally, an RTG is a 4-tuple $G = (S, \mathcal{N}, F, R)$ consisting of an axiom S, a finite set $\mathcal{N}$ of zero-arity non-terminal symbols with $S \in \mathcal{N}$, a set $\mathcal{F}$ (disjoint with $\mathcal{N}$) of terminal symbols each having a fixed arity, and a finite set $\mathcal{R}$ of production rules of the form $A \rightarrow \beta$, with A a non-terminal of $\mathcal{N}$ and β a *term* over $\mathcal{F} \cup N$. A term over some set of fixed-arity symbols M is defined recursively as a symbol of M applied to n arguments with n equal to the arity of the symbol and each of the arguments being a term over M. A set of fixed-arity symbols is also called a *ranked alphabet*, and the set of terms over the ranked alphabet M is written $T(M)$.

The language described by an RTG is a *regular tree language*. A given RTG $G = (S, \mathcal{N}, \mathcal{F}, \mathcal{R})$ describes the language consisting of all terms t over $\mathcal{F}$ such that the axiom S can be rewritten as t via a series of rewrites licensed by the rules in $\mathcal{R}$. In other words, to derive a term of the language, we start from the axiom and apply rules until we have a term containing no non-terminal symbols.

More formally, the derivation relation $\rightarrow_G$ associated to G is a relation on pairs of terms of $T(\mathcal{F} \cup \mathcal{N})$ such that $s \rightarrow_G t$ if and only if there is a rule $A \rightarrow \alpha \in \mathcal{R}$ such that substituting α for an instance of A in s gives t; and the language generated by G, denoted by $L(G)$, is $\{s \in T(\mathcal{F}) \mid S \rightarrow_G^+ s\}$ with $\rightarrow_G^+$ the transitive closure of $\rightarrow_G$. The subscript G on the symbols $\rightarrow_G$ and $\rightarrow_G^+$ can be omitted if the grammar is clear from the context.

As is well known ([Vijay-Shanker *et al.*, 1987; Shieber, 2006]), RTG can be used to generate the derivation trees licensed by a TAG grammar. Intuitively, the RTG representation of a TAG elementary tree is a rule that rewrites the requirement satisfied by that tree as a local tree whose root is the tree name and whose leaves are the introduced requirements. A substitution / adjunction requirement for a tree of root category X is written as X_S and X_A, respectively.

Figure 3.4 shows in the right side the rules of an example RTG which describes the derivation trees of the toy TAG grammar depicted in the left part of the figure. The RTG terminals ({ *the, man, runs, often,* ϵ}) refer to the elementary trees of the TAG grammar while its non-terminals ($\{NP_S, S_S, NP_A, VP_A, V_A, S_A, Det_A\}$) describe the adjunction and substitution requirements that can be introduced by an elementary tree. Further, each elementary tree t in the input TAG gives rise to an RTG rule whose left hand side (lhs) expresses the syntactic requirement that t can satisfy and whose right hand side (rhs) expresses the syntactic requirements it



r1.	NP_S	$\rightarrow$	$man(NP_A)$
r2.	S_S	$\rightarrow$	$runs(S_A\ NP_S\ VP_A\ V_A)$
r3.	VP_A	$\rightarrow$	$often(VP_A)$
r4.	NP_A	$\rightarrow$	$the(NP_A\ Det_A)$
r5.	NP_A	$\rightarrow$	ϵ
r6.	S_A	$\rightarrow$	ϵ
r7.	V_A	$\rightarrow$	ϵ
r8.	VP_A	$\rightarrow$	ϵ
r9.	Det_A	$\rightarrow$	ϵ

Figure 3.4: Example RTG describing the derivation trees of a toy TAG.

introduces. If the tree is an auxiliary tree, it can satisfy an adjunction requirement and the category labelling the lhs of the RTG rule is subscripted with A. If it is an initial tree, the lhs category of the RTG rule is subscripted with S to indicate that it can satisfy a substitution requirement. Further, each node in the elementary tree which either requires a substitution or allows for an adjunction introduces a daughter node in the rhs RTG term whose category reflects the allowed/required adjunction/substitution. To capture the fact that adjunction is optional, there are additional rules allowing any adjunction requirement to be rewritten as the symbol ϵ, a terminal symbol of the RTG.

We just saw how to map a TAG to an RTG of TAG derivations. Schmitz and Le Roux (2008) further extend this mapping to FB-TAG as follows. In the resulting FB-RTG, each non-terminal symbol on the left and right side of a rule is marked up with a feature structure with **top** and **bottom** attributes. For a symbol on the right side, the values of those attributes are equal to the top and bottom feature structures of the corresponding TAG tree node (substitution node or adjunction site). For the symbol on the left, they are the *interface* of the tree to any node into which the tree is inserted. When an initial tree is inserted into a substitution node of another tree, its root node's top unifies with the substitution node's top. Thus, the interface of the initial tree is its root node's top, and this appears as the **top** attribute of the symbol on the left side of the corresponding RTG rule. For an auxiliary tree, the interface is the top of the root node and the bottom of the foot node (cf. Section 2.2.2), so these appear as the **top** and **bottom**, respectively, of the left side of the corresponding rule.

To run some examples, let's take up again the example grammar in Figure 3.2 Section 3.1.1 but this time we build the *SemTAG* version (i.e. an LTAG with feature structures and semantics). The resulting grammar is shown in Figure 3.5.

There will generally be co-indexed feature values in a rule. In a substitution rule

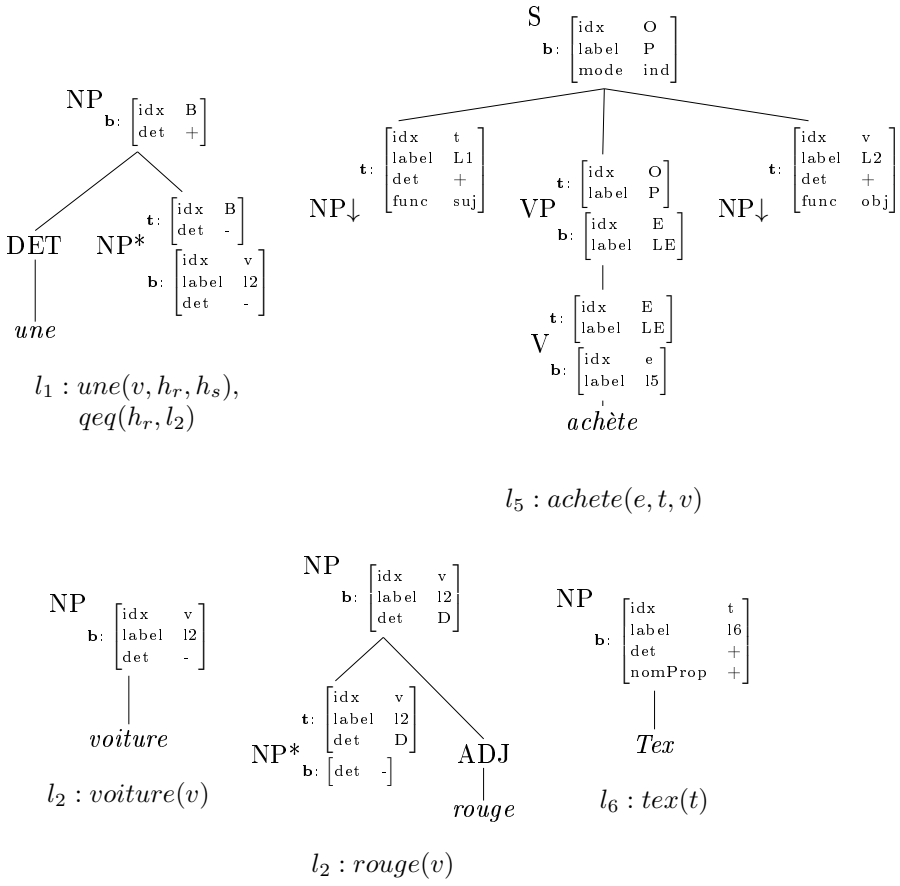


Figure 3.5: An example SemTAG sub-grammar selected for the input $\{l_1 : une(v, h_r, h_s), qeq(h_r, l_2), l_2 : voiture(v), l_2 : rouge(v), l_5 : achete(e, t, v), l_6 : tex(t)\}$ corresponding to the sentence *Tex achète une voiture rouge* (Tex buys a red car). Note: capital letters represent variable values (underspecified feature values).

such as (22), which is a translation of the *Tex* tree from Figure 3.5, the **top** value of the left side symbol is equal to the top feature structure of the root node of the tree, and therefore is co-indexed with the **top** value of the right side symbol that embodies that node. In an epsilon rule such as (23), the **top** and **bottom** values on the left side are co-indexed with each other to enforce the requirement that the top and bottom feature structures of each node in the derived tree must unify (cf. Section 2.2.2).

(22)

$$NP_S[\mathbf{top}\ \boxed{1}] \rightarrow Tex(NP_A\begin{bmatrix}\mathbf{top} & \boxed{1}\ \text{T} \\ \mathbf{bottom} & \begin{bmatrix}\text{idx,label} & \text{t,l6} \\ \text{det} & + \\ \text{nomProp} & +\end{bmatrix}\end{bmatrix})$$

(23)

$$NP_A\begin{bmatrix}\mathbf{top} & \boxed{1}\\ \mathbf{bottom} & \boxed{1}\end{bmatrix} \rightarrow \epsilon$$

r1. $NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{v}\\ \text{label} & \text{l2}\\ \text{det} & \text{-}\end{bmatrix} & \end{bmatrix} \rightarrow une(\ NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{v}\\ \text{det} & +\end{bmatrix} & \end{bmatrix}\ Det_A\ )$

$l_1 : une(v, h_r, h_s), geq(h_r, l_2)$

r2. $S_S\left[\mathbf{t}\ \boxed{1}\right] \rightarrow achete(\ S_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{O}\\ \text{label} & \text{P}\\ \text{mode} & \text{ind}\end{bmatrix} & \end{bmatrix}\ NP_S\begin{bmatrix}\mathbf{t}\begin{bmatrix}\text{idx} & \text{t}\\ \text{label} & \text{L1}\\ \text{det} & +\\ \text{func} & \text{suj}\end{bmatrix} & \end{bmatrix}\ VP_A\begin{bmatrix}\mathbf{t}\begin{bmatrix}\text{idx} & \text{O}\\ \text{label} & \text{P}\end{bmatrix} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{E}\\ \text{label} & \text{LE}\end{bmatrix} & \end{bmatrix}$

$V_A\begin{bmatrix}\mathbf{t}\begin{bmatrix}\text{idx} & \text{E}\\ \text{label} & \text{LE}\end{bmatrix} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{e}\\ \text{label} & \text{l5}\end{bmatrix} & \end{bmatrix}\ NP_S\begin{bmatrix}\mathbf{t}\begin{bmatrix}\text{idx} & \text{v}\\ \text{label} & \text{L2}\\ \text{det} & +\\ \text{func} & \text{obj}\end{bmatrix} & \end{bmatrix})$

$l_5 : achete(e, t, v)$

r3. $NP_S\left[\mathbf{t}\ \boxed{1}\right] \rightarrow voiture(\ NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{v}\\ \text{label} & \text{l2}\\ \text{det} & \text{-}\end{bmatrix} & \end{bmatrix})$

$l_2 : voiture(v)$

r4. $NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1} & \\ \mathbf{b}\left[\text{det}\ \ \text{-}\right] & \end{bmatrix} \rightarrow rouge(\ NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T} & \\ \mathbf{b}\begin{bmatrix}\text{idx} & \text{v}\\ \text{label} & \text{l2}\\ \text{det} & \text{D}\end{bmatrix} & \end{bmatrix}\ Adj_A\ )$

$l_2 : rouge(v)$

r5. $NP_S\left[\mathbf{t}\ \ \boxed{1}\right] \rightarrow Tex(NP_A\begin{bmatrix}\mathbf{t}\ \ \boxed{1}\ \text{T} & \\ \mathbf{b}\begin{bmatrix}\text{idx,lbl} & \text{t,l6}\\ \text{det} & +\\ \text{nomProp} & +\end{bmatrix} & \end{bmatrix})$

$l_6 : tex(t)$

r6. $NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{P} & \\ \mathbf{b}\ \boxed{1}\ \text{P} & \end{bmatrix} \rightarrow \epsilon$ **r7.** $S_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{P} & \\ \mathbf{b}\ \boxed{1}\ \text{P} & \end{bmatrix} \rightarrow \epsilon$

Figure 3.6: FB-RTG translation of the SemTAG sub-grammar shown in Figure 3.5

Figures 3.5 and 3.6[15] illustrate the FB-TAG to FB-RTG conversion explained in previous paragraphs.

The derivation relation defined for FB-RTG makes use of combinations of rewrites and unifications of terms. The FB-RTG defined by Schmitz and Le Roux (2008)

derivation relation

[15]Note that in the translation from the tree for *une* the indew variable B is translated with the value v, this is because when the foot node contains a top feature structure it is kept for further unification, here for simplicity we already take the corresponding value for the variable B.

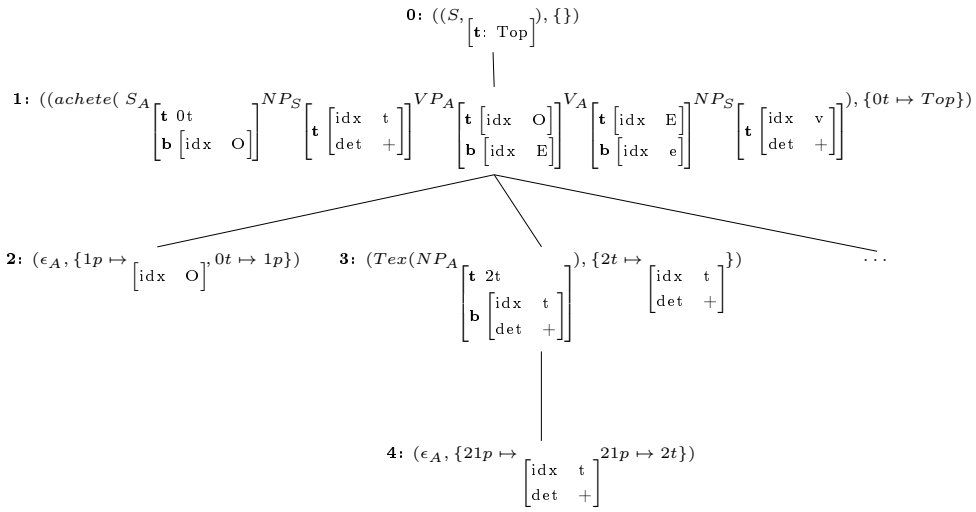
is a 5-tuple $G = (S, \mathcal{N}, \mathcal{F}, \mathcal{D}, \mathcal{R})$, where $S, \mathcal{N}, \mathcal{F}, \mathcal{R}$ are defined as in the RTG version. In addition, an FB-RTG includes an extra set, that is the set $\mathcal{D}$ of features structures; and the rules in the set $\mathcal{R}$ are of the form, as illustrated previously by examples, $(A, d) \rightarrow a((B_1, d'_1), ..., (B_n, d'_n)))$ where $B, B_1, ..., B_n$ are non-terminals and $d, d'_1, ..., d'_n$ are features structures. The derivation relation of an FB-RTG G $\rightarrow_G$ is defined (cf. [Schmitz and Le Roux, 2008]) as follows. Two pairs of terms from $T(\mathcal{F}, \mathcal{N} \times \mathcal{D})$ and their set of substitutions stand in a derivation relation $(s, e) \rightarrow_G (t, e')$ iff: $\exists$ context C in s and rule $(A, d) \rightarrow a((B_1, d'_1), ..., (B_n, d'_n)))$ in $\mathcal{R}$ with fresh variables in the feature structures, a structure d', and a unification σ verifying:

$$s = C[(A, d')],\ t = C[a((B_1, \sigma(d'_1)), ...(B_n, \sigma(d'_n)))],$$

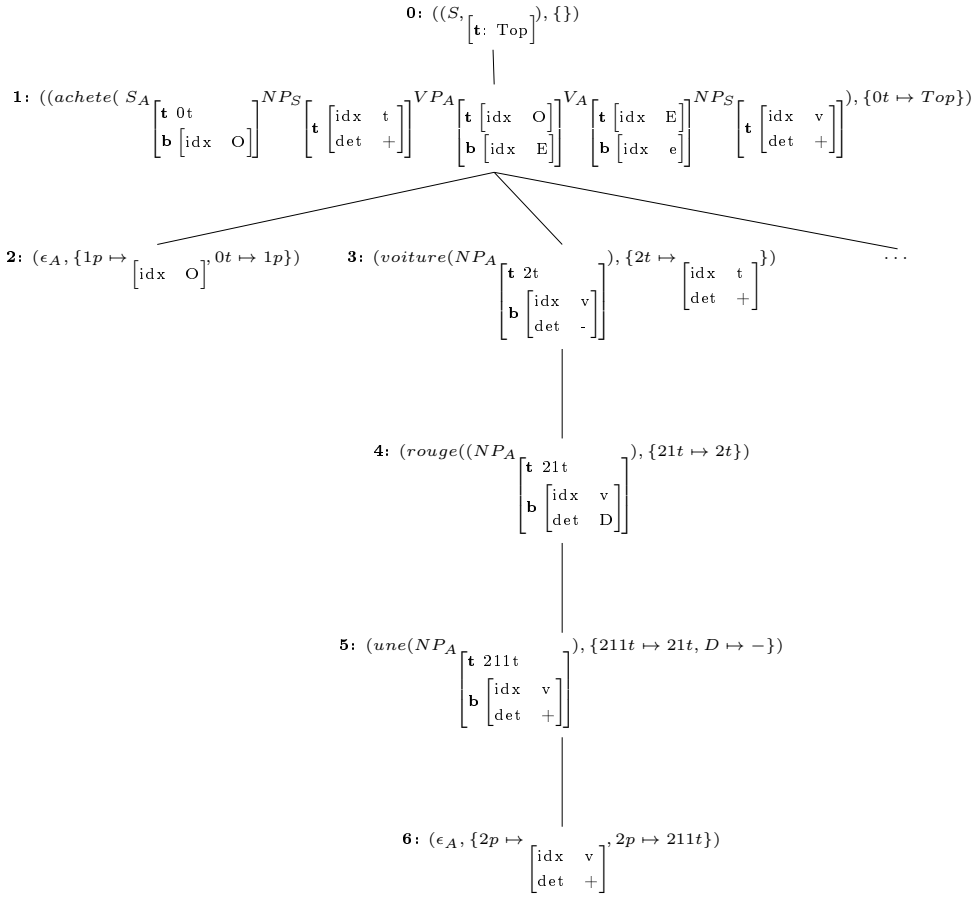
$$\sigma = mgu(d, e(d')) \text{ and } e' = \sigma \circ e$$

Below, we follow some steps of an example FB-RTG derivation with the grammar in Figure 3.6[16] to illustrate the working of the feature structure unifications and how each derivation step embodies a tree combination operation (substitution or adjunction of the original TAG grammar). We start the derivation from the initial symbol $S = S_S$ and apply rule $r2$ (derivation steps (0:) and (1:)). The step (2:) expands the first right-hand side non-terminal of (1:), i.e. S_A, using the epsilon rule $r7$, this step is equivalent to a top and bottom feature structures unification of a node in TAG terms. To expand the next non-terminal of (1:), NP_S, there are two candidate rules: $r3$ and $r5$. We choose the rule $r5$, corresponding to the tree for *Tex*. Then, the NP_A non-terminal in (3:) is further expanded using the epsilon rule $r6$. When unifying the left-hand side of $r6$ with NP_A in (3:), the top and bottom feature structures are unified. As all unification steps succeed the partial derivation succeeds, but it is only in step (4:) that we know that the choice of the rule $r5$ was the right one.

[16] We omit some features "label, nomProp, mode and func" to simplify the example



If on the contrary, we would have chosen the sequence of rewriting steps using rules $r3, r4, r1, r6$, the derivation would have failed as we would have encountered a unification failure in the last step (applying epsilon rule $r6$) due to the feature `idx`. Below, the unsuccessful derivation steps:



The derivation in the second NP_S of (1:) involves the application of the sequence of rules $r3, r4, r1, r6$. Note that in step (i+1:), if we apply rule $r6$ we obtain a feature unification failure (i.e. in TAG terms, if after substituting the tree *voiture* into the tree *achete* we try to unify the top and bottom features of the node where substitution took place). Similarly, if we would have tried the derivation sequence $r3, r1, r4, r6$, we would have found a feature unification failure (i.e. again, in TAG terms, if we look the corresponding elementary trees in the FB-TAG grammar in Figure 3.5, the successful derivation sequence is adjoining the tree for *rouge* into the tree for *voiture*).

(24)

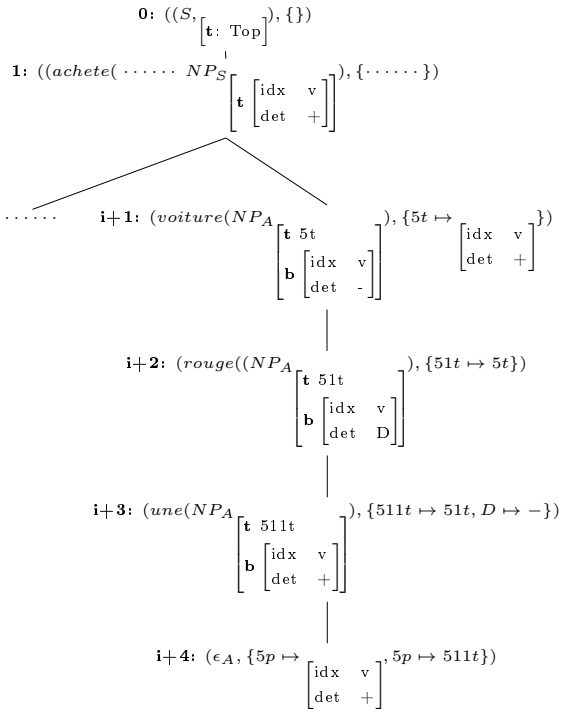


Figure 3.7 shows a complete derivation for the sentence *Tex achète une voiture rouge* (Tex buys a red car) with the FB-RTG grammar of Figure 3.6.

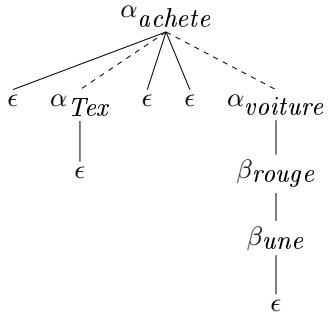


Figure 3.7: FB-RTG derivation.

Schmitz and Le Roux (2008) argue that derivations in FB-RTG are not very predictive, because substitution sites are only fully defined once all adjunctions have taken place. Figure 3.8a shows a derivation tree for the sentence *One of the cats has caught a fish* with the normal FB-RTG translation that exemplifies their point. That is, the subject/verb agreement in the main verb node is only determined in step (5:), i.e. after all adjunctions have taken place. More generally, Schmitz and Le Roux's observation is that substitution sites admit their expansion by most of the initial trees because their root top feature structure is often underspecified (e.g. trees for *voiture* and *Tex* in FB-TAG grammar in Figure 3.5).

left-corner transformed grammar

Thus, Schmitz and Le Roux propose two ways of encoding an FB-TAG. The one we have seen so far, i.e. FB-RTG, and another variant which is a *left corner* transformation, viz. left-corner FB-RTG. This transformation enforces derivations in a different order: adjunctions at root nodes are applied first, i.e. before the initial tree substitution. Figure 3.8b illustrates such a derivation for the same sentence. In this case, information about subject/verb agreement becomes available earlier, i.e. after the first adjunction.

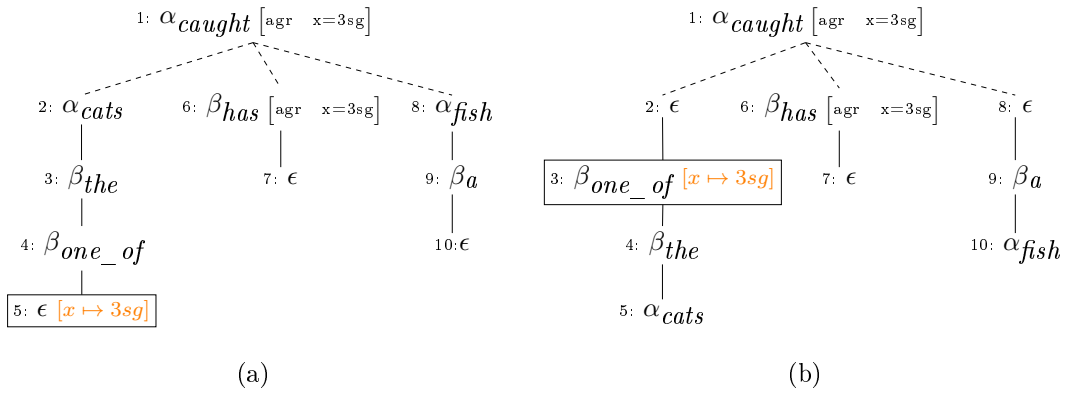
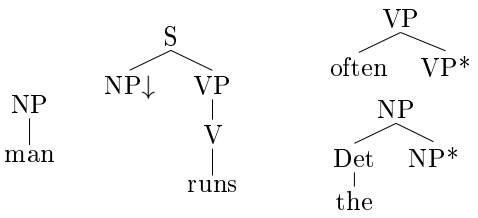


Figure 3.8: FB-RTG derivation tree (a.) and left-corner FB-RTG derivation tree (b.) for the sentence *One of the cats has caught a fish*. Node labels of the derivation trees start with αs and βs indicating whether they correspond to an initial or auxiliary tree respectively.

To obtain derivations in reversed order, initial trees and auxiliary trees that adjoin at root positions are translated in a slightly different way[17]. An example of a transformed RTG grammar is shown in Figure 3.9. New ε-rules of rank 1 performing top/bottom unifications and rewriting only non-terminals subscripted with S, e.g. NP_S, are introduced. Root nodes of initial trees are not present in the right-hand side of the corresponding FB-RTG rule, because there will be no root adjunction

[17] Auxiliary trees that do not occur at the root are translated in the same way, auxiliary trees that occur at root positions are translated in both ways.



r1.	NP	$\rightarrow$	man
r2.	S	$\rightarrow$	$runs(NP_S\ VP_A\ V_A)$
r3.	VP_A	$\rightarrow$	$often(VP_A)$
r4.	NP_A	$\rightarrow$	$the(NP_A\ Det_A)$
r5.	NP	$\rightarrow$	$the(NP\ Det_A)$
r6.	NP_S	$\rightarrow$	$\epsilon(NP)$
r7.	V_A	$\rightarrow$	ϵ
r8.	NP_A	$\rightarrow$	ϵ
r9.	VP_A	$\rightarrow$	ϵ
r10.	Det_A	$\rightarrow$	ϵ

Figure 3.9: Example of left-corner transformed RTG describing the derivation trees of a toy TAG (the same as that of Figure 3.4).

taking place at root nodes of initial trees. We can think of the left-corner translated auxiliary trees that adjoin into root positions as playing the role of initial trees where the foot node acts as a sort of substitution node.

The features in the transformed RTG are translated in a different way too (cf. [Schmitz and Le Roux, 2008] for a formal definition about the translation of the features in the left-corner transformation). In Figure 3.10, we recall some of the elementary trees of the FB-TAG example grammar in Figure 3.5. In Figure 3.11, we show the left-corner FB-RTG rules corresponding to those trees in Figure 3.10. For FB-RTG rules from initial trees, the lhs top and bottom features are the top and bottom features of the root node of the corresponding initial tree. For FB-RTG rules from auxiliary trees adjoining into root nodes, the lhs takes the features from the root node of the auxiliary tree and the "left-most" non-terminal of the right-hand side contains those feature structures that make up the interface of the tree (i.e. those feature structures that are involved in adjunctions: top of the root node and bottom of the foot node).

In the present section, we have explained the FB-TAG to FB-RTG translation. We described how TAG elementary trees are encoded into RTG rules including feature structures. In our FB-TAG grammar, elementary trees are associated with a unification-based compositional semantics and tree nodes are decorated with unification variables from that associated semantics (cf. Section 2.2.4). FB-RTG rules other than the epsilon rules are also associated with a semantic formula containing unification variables. These are carried over as is from the FB-TAG trees to the FB-RTG rules. This is possible because feature structures are preserved and so are those features corresponding to the unification variables from the associated semantic formula (e.g. `idx` and `label`).

In the following sections, we present the core derivation tree generation algorithm

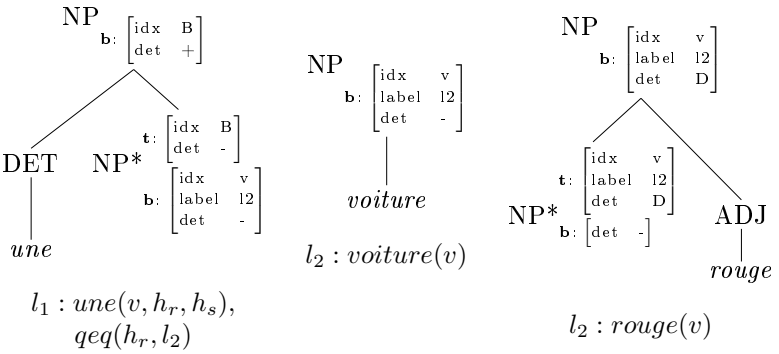


Figure 3.10: Recall of elementary trees for *une*, *voiture*, *rouge* from the grammar in Figure 3.5

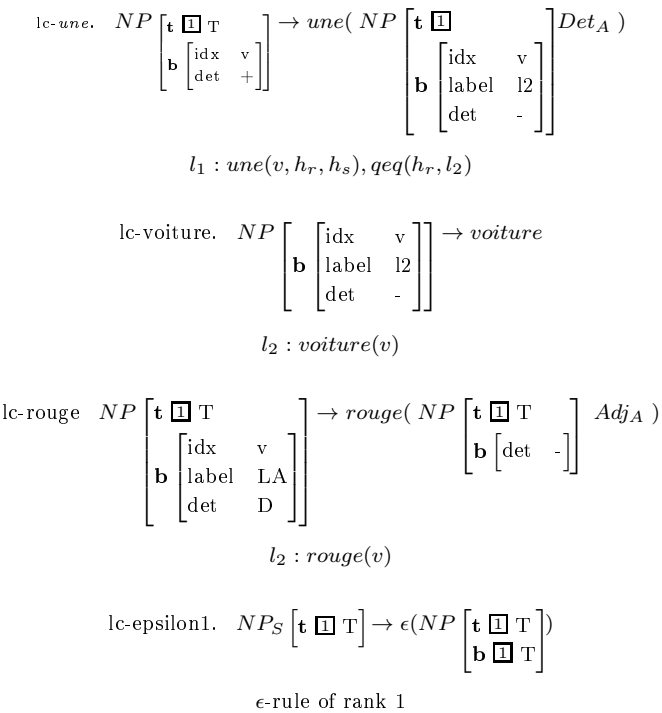


Figure 3.11: Left-corner FB-RTG translation of the trees *voiture*, *rouge une* of the SemTAG grammar fragment shown in Figure 3.10

based on the FB-RTG encoding[18] of FB-TAG.

[18]Both the original and the left-corner transformed one.

3.2 RTGen surface realisation algorithm

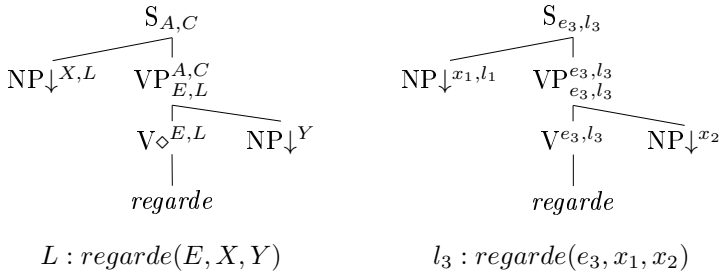
SemTAG (the FB-TAG for English and French described in detail in Section 2.2) is a wide-coverage reversible grammar which associates natural language expressions with syntactic trees and meaning representations. When used for generation, the input is a meaning representation (flat semantics representation of the form discussed in 2.2.4) and the output is the set of strings associated by the grammar with the given meaning. Furthermore, in the same way that lexicalised grammars permit stating a parsing algorithm in two major stages: sub-grammar selection and structure combination (cf. Section 2.2.2), they also enable this division in generation. The difference being that instead of selecting structures from the grammar for lexical items in the input string; in generation, we select grammar structures associated with semantic representations from the input semantics. Thus, our *SemTAG* based surface realisation algorithm (similarly to existing ones, e.g. [Gardent and Kow, 2007; Koller and Striegnitz, 2002]), starts from a meaning representation and pipelines three main phases:

- **Lexical selection** selects from the grammar those elementary trees whose semantics subsumes part of the input semantics.
- **Tree combination** systematically tries to combine trees using substitution and adjunction.
- **Retrieval** unpacks the generation forest and extracts the yields from complete derivation trees, thereby producing the generated sentence(s).

Given an input semantics ϕ, the lexical selection step will select those items from the grammar, whose semantic representation ψ unifies with part of the input semantics ϕ. For instance, given the input semantics in (25-a), the lexical items in Figure 3.12 will be selected. We recall from Section 2.2.4, that the unification variables occurring in the lexical semantics also occur in the feature structures of the elementary trees. After a lexical item is selected (unification of lexical semantics and part of the input semantics), the semantic indices of the input semantic formulae are propagated in the elementary trees' feature structures.

(25) a. $\phi = \{l1 : tammy(x_1),\ l2 : tex(x_2),\ l_3 : regarde(e_3, x_1, x_2),\ l_3 : souvent(e_3)\}$
b. *Tammy regarde souvent Tex.* (Tammy often looks at Tex)

The RTGen algorithm manipulates RTG rules describing the contribution of the SemTAG elementary trees to the derivation tree rather than the elementary tree themselves. In what follows, we describe the other two steps of surface realisation, namely tree combination and retrieval (i.e. unpacking and string extraction).



$L : regarde(E, X, Y)$ $\quad$ $l_3 : regarde(e_3, x_1, x_2)$

Figure 3.12: The lexical item in the left is selected given the input semantics in (25), $\{L : regard(E, X, Y)\} \sqsubseteq \psi$. Note that e_3, x_1, x_2 are constants. Thus, in the generation process, x_1 would never be instantiated with x_2 or any other constant.

3.2.1 RTGen's base algorithm

Given an FB-RTG encoding G_{FB-RTG} from the sub-grammar G_{FB-TAG} resulting from the lexical selection phase, the algorithm described in [Perez-Beltrachini, 2009] computes $L(G_{FB-RTG})$, i.e., the generation forest.

This base algorithm ([Perez-Beltrachini, 2009]) integrates several ideas and techniques from the parsing literature. It draws on Pereira and Warren (1983) and Shieber *et al.*'s (1995) deductive parsing framework for unification-based grammatical formalisms. To avoid the repeated computation of intermediate structures common to several larger parse structures, it integrates Kay's (1986) chart mechanism. Finally, the adopted parsing strategy is an Earley style algorithm ([Earley, 1970]) adapted to support generation from a flat semantics.

In essence, RTGen implements a chart-based Earley-style algorithm for the FB-RTG encoding of *SemTAG*. Table 3.2 sketches the Earley-style generation algorithm stated as a deduction system ([Shieber *et al.*, 1995]). The standard *item representation* is the pair $[(A, d) \rightarrow T_a((B_1, d_1), .., \bullet(B_i, d_i), ..., (B_n, d_n)), \psi]$. In the first component, the dot in the production marks the point reached in the generation of the derivation tree. The non-terminal symbols (B_i, d_i) in the dotted rule are complex non-terminals from the FRTG rules (i.e. a non-terminal symbol, syntactic category and operation type, B_i and a feature structure d_i). T_a is the ranked terminal of the FRTG rule (i.e. is the elementary tree family). The second component of the item, ψ, is a flat semantic formula. In the items, we do not keep track of string positions, as is usually done when parsing a string, but rather we keep the associated semantic formula. The algorithm starts from the initial fact, the *axiom*, $[S' \rightarrow \bullet S_S, \emptyset]$. Note that in this item the non-terminal symbol S_S is the axiom in the FRTG grammar while the second component represents the empty semantics. As we are generating from an input semantics (i.e. the semantic input to the realiser), the subset of

the input semantics analysed so far is empty. On the other hand, in the *goal* item $[S' \rightarrow S_S\bullet, \phi]$ the dot at the end of the item production means that the whole derivation tree with root S_S has been traversed. At this point, the semantics ϕ should be exactly the input semantics. Further, inference rules are associated with a set of constraints (*side conditions* [Shieber *et al.*, 1995]). In Table 3.2, "where"-statements describe constraints regarding semantic coverage.

Table 3.2: RTGen derivation tree generation algorithm (deductive system).

Axiom	$\overline{[S' \rightarrow \bullet S_S, \emptyset]}$
Goal	$[S' \rightarrow S_S\bullet, \phi]$ where ϕ is the input semantics.
Prediction	$\frac{[(A,d) \rightarrow T_a(\alpha \bullet (B,d_i)\ \beta),\ \varphi]}{[(B,\sigma(d')) \rightarrow T_b(\bullet(B_1,\sigma(d'_1)),...,(B_n,\sigma(d'_n))),\ \psi]}$ where $(B,d') \rightarrow T_b((B_1,d'_1),...,(B_n,d'_n))$ is a rule in the grammar with associated semantics ψ, $\sigma = mgu(d_i,d')$ and $\varphi \cap \psi = \emptyset$ α, β are sequences of non-terminals
Completion	$\frac{[(A,d) \rightarrow T_a(\alpha \bullet (B,d_i)\ \beta),\ \varphi]\ [(B,d') \rightarrow T_b(\rho \bullet),\ \psi]}{[(A,\sigma(d)) \rightarrow T_a(\alpha\ (B,\sigma(d_i))\ \bullet\ (C,\sigma(d_{i+1}))\ \delta),\ \xi]}$ where $\sigma = mgu(d_i,d')$, $\varphi \cap \psi = \emptyset$ and $\varphi \cup \psi = \xi$ $\alpha, \beta, \rho, \delta$ are sequences of non-terminals

The deduction procedure implemented in RTGen follows from the agenda-driven, chart-based deduction procedure proposed in [Shieber *et al.*, 1995]. Its generic steps are:

1. Initialise the chart as an empty set of items and the agenda with the axiom items.
2. Repeat the following steps until the agenda becomes empty:
   - Select an item from the agenda *trigger item*.
   - Add the item to the chart (if not already contained in)
   - Generate all the items that are immediate consequences of the trigger item and the items in the chart. Add the new items to the agenda.
3. If the goal item is in the chart, then the goal is proved; otherwise it is not.

In what follows, we discuss the techniques included in the RTGen derivation

tree generation algorithm ([Perez-Beltrachini, 2009; Gardent and Perez-Beltrachini, 2010]).

Chart based generation

While applying the inference rules we want the derived items (or consequences) to be "cached", so they can be re-used instead of being re-computed when needed again in future computations. Tabulation techniques are motivated by considering problems which display a high degree of redundant computations and aim at keeping a cache of the computed elements. A *chart* data structure in chart-based parsing algorithms ([Kay, 1986]) provides such a cache mechanism. One major cause for redundancy in Natural Language parsing (and generation) lies in the inherent ambiguity of natural language and thereby of its grammar. In particular, in our case, because the grammar used is a wide-coverage lexicalised grammar, lexical ambiguity (the number of grammatical units associated with each word or lexical semantics) is very high. Indeed, for one lexical item there might be several families with several trees. Furthermore, we might choose to explore not just one but all possible solutions. Hence, this process generates several intermediate results which are used to build several distinct derivations.

Subsumption based blocking of new items. As a result of applying the inference rules, new items are produced and we want them to be stored in the chart. However, we need to verify that these new items have not already been produced. This verification, in the RTGen algorithm, involves subsumption checking ([Shieber *et al.*, 1995]). The reason is that the RTG grammar rules contain feature structures with underspecified features values (cf. Figure 3.5), and sometimes underspecified syntactic category (as in the case of epsilon rules). Therefore, newly derived items may differ only in terms of instantiations. Instead of keeping every differently instantiated item, the idea is to keep only the most general form which would be available in the chart for further utilization in different operations. Thus, we have to check whether a more general item has already been stored in the chart. In short, a new item should be added to the chart only if no subsuming (more general) item already exists in it.

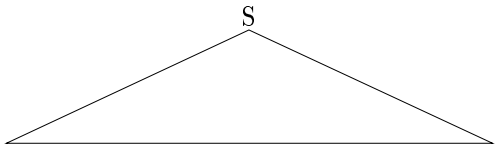
Agenda based control. To control the application of the inference rules, we have choosen an agenda-driven chart algorithm (cf. previous section). Derived items are not directly placed into the chart but stored in the *agenda* (i.e. an auxiliary storage structure). The use of an agenda allows the customization of the search strategy.

Specifically, implementing the agenda as a stack would provide a depth-first search strategy, whereas implementing it as a queue would result in a breadth-first search. The difference in the search strategy does not affect the results produced by the selected tree traversal. In the RTGen algorithm the agenda is a stack.

Redundancy checking in the agenda. As pointed out by Shieber *et al.* (1995), we can carry out a step in advance by verifying the existence of derived items when they are added into the agenda.

Packed Shared Generation Forest

A packed shared forest ([Billot and Lang, 1989]) is a compact representation of a potentially huge search space. In parsing, a packed forest is the compact representation of all possible hierarchical structures covering the same phrase string. In contrast, in generation (Figure 3.13), a packed forest is a compact representation of one or few hierarchically (different) structures covering the same string plus several different structures with their corresponding strings covering the same input semantics ([Langkilde, 2000; Carroll and Oepen, 2005]).



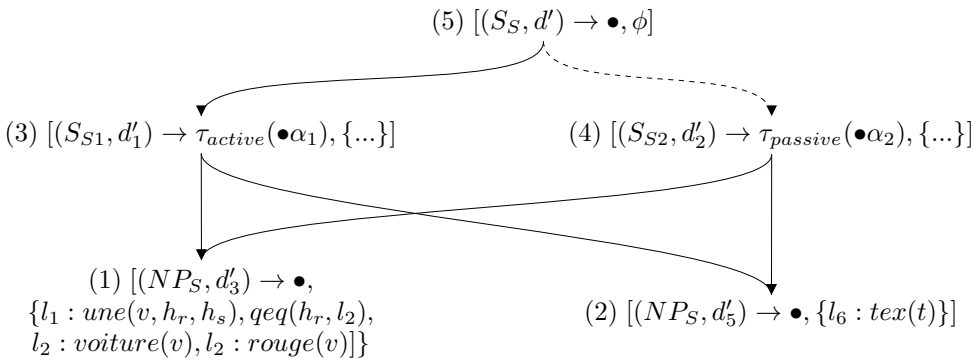
Tex achète une voiture rouge
Une voiture rouge est achètée par Tex
$\{l_1 : une(v, h_r, h_s), qeq(h_r, l_2), l_2 : voiture(v),$
$l_2 : rouge(v), l_5 : acheter(e, t, v), l_6 : tex(t)\}$

Figure 3.13

To construct a packed shared generation forest, the RTGen algorithm relies on two concepts: (i) the "offline" generation of derivation trees ([Shieber *et al.*, 1995]) and (ii) "item equivalence", i.e. subsumption-based checking of new edges with identical semantic coverage ([Carroll and Oepen, 2005]).

Information about how a given item in the chart was obtained (i.e. derivation history) is not stored in the chart together with the item. Instead, the derivation history is separately represented by means of pointers towards chart items. At the end of the generation process, the generation forest represents a set of derivation trees, which could be extracted in a subsequent step.

Figure 3.14 shows an excerpt of the chart data structure for the generation from the semantic input ϕ. The arrows illustrate pointers from consequent items to an-



$$\phi = \{l_1 : une(v, h_r, h_s), qeq(h_r, l_2), l_2 : voiture(v), l_2 : rouge(v), l_5 : acheter(e, t, v), l_6 : tex(t)\}$$

Figure 3.14: Example of items in a chart (excerpt) and generation forest for the generation from ϕ of the sentences *Tex achète une voiture rouge* and *Une voiture rouge est achètée par Tex*.

tecedent items. For instance, items 1 and 2 are used in the derivation of items 3 and 4 –sharing–, further in the derivation, items 3 and 4 will be merged into item 5 –packing.

This packed shared forest mechanism comes almost for free from Earley-style chart-based deduction framework, pointers and subsumption checking of new items. Though, the compactness of the forest could be further improved.

As aforementioned, the verification whether an item already exists in the chart consists in checking if there exists in the chart a more general item. If such item does exist, the new item is considered as already existing and packed together with the item already present in the chart. However, if the new item happens to be more general than the one in the chart, the new item is also added to the chart without packing taking place. This mechanism is called *proactive packing* (cf. [Oepen and Carroll, 2000]). An alternative solution in this case would be to replace the more specific chart item by the newer more general, and then packing them together. But this implies that previous derivations involving the existing more specific item need to be revised. This alternative mechanism is called *retroactive packing* by Oepen and Carroll (2000). In RTGen's algorithm we have implemented proactive packing which is the straightforward solution. Nevertheless, it would be interesting to investigate how RTGen algorithm could be modified to integrate retroactive packing.

In this packing schema, active items of the form $[X \rightarrow \bullet\alpha Y]$ and $[X \rightarrow \bullet\beta Y]$ might be also packed into a single entity when reaching a derivation point at an item of the form $[X \rightarrow \bullet Y]$. However, differences in α and β could have been factored out by defining a more "abstract" equivalence criteria considering only certain in-

formation present on them. This can be achieved by applying *restrictors* ([Oepen and Carroll, 2000]) on the grammar rules. A restrictor that removes most of the grammatical features would result in a very compact generation forest representation and in a more laborious unpacking phase. The appropriate choice of restrictors determines the optimality of packing.

Indexing

In string parsing, chart items standardly contain two indices, pointing at the start and end positions of the recognised span over the input string. These positions help not only in ensuring correctness but also in improving efficiency, as two edges are only considered for combination whenever they are adjacent. In this way, the number of non-productive attempts at applying the inference rules is reduced. In our problem, because the input to surface realisation is a flat semantic formula, there are no string positions to use. Instead, we use the information given by the semantic indices in the semantic formulae as proposed in [Kay, 1996; Carroll *et al.*, 1999].

Based on these proposals, RTGen implements the following mechanism for indexing items in the chart. Each item is associated with two semantic indexes that we called *index* and *dotted-index*. Moreover, we distinguish between active and passive items. For active items, the index *index* is the semantic index associated with the left-hand side of the rule and the *dotted-index* is the semantic index associated with the active symbol (symbol after the dot) in the right-hand side. For passive items, we require both indices to coincide, that is we the *dotted-index* has the same value as the *index*, in fact, as the item is passive there is no active symbol.

Semantic filter. A semantic filter is implemented based on the standard idea of item coverage ([Kay, 1996; Carroll *et al.*, 1999]). In our algorithm, we use this as described in the previous section (preconditions regarding semantics in the inference rules). First, the semantic filter in item prediction permits the prediction of new edges only if the semantic of the newly predicted item is disjoint with the semantic of the active one. This means that, we only predict items that would successfully combine with others, at least from the semantic coverage point of view. Second, the semantic filter in the item combination takes place allowing or not those item combinations. That is, checking if their semantic does not cover the same lexical items. When edges are combined their semantics are too.

3.2.2 Extensions to the base algorithm

Further exploiting semantic indices

The indexing mechanism described in the last section, though useful in some cases, is not always effective. As we have seen in the FB-RTG example grammar in Figure 3.6 (Section 3.1.3 p.54), for some non-terminal symbols, the index feature (`idx`) value is underspecified (i.e. a unification variable that has not yet being instantiated). In general, this corresponds to non-terminal symbols in the righ-hand side of the rules stemming from internal phrasal nodes of the FB-TAG elementary trees. But also the value of the `idx` feature might be under-specified for non-terminal symbols of the left-hand side of the rules.

Therefore, we extend the indexing mechanism to use the semantic information, i.e. the semantic indices, present in the semantic predicates associated with each rule to better guide the prediction and completion operations. We combine information about semantic indices (distinguished index and argument indices) in the predicates of the input semantics ([Copestake, 2008; Copestake, 2009]) with information about the type of the non-terminals (representing substitution or adjunction sites).

In prediction from an active item of the form $(A, d) \rightarrow \bullet(B, d_1)\alpha$ with semantics $rel(x_1, x_2, .., x_n)$, only new items stemming from rules whose left-hand side unifies with the non-terminal symbol (B, d_1) (as stated before, Table 3.2) and whose associated semantics has some index belonging to the set $x_1, x_2, ..., x_n$ will be created. During prediction this constraint helps in reducing the work carried out. It might happend that the `idx` features in the left-hand side of the FB-RTG rules are underspecified (see for instance, rules $r3.$ and $r4$ corresponding to initial trees in Figure 3.6 p.54).

For completion, given a semantic predicate of the form $rel(x_1, x_2, .., x_n)$ associated to some item, we differentiate two sets of semantic indices. One, is the single set that we call *offered index* comprised of the distinguished index ([Copestake, 2008; Copestake, 2009]), i.e. x_1. The other, is the set of argument indices that we call *required indices*, i.e. $x_2, ..., x_n$. Given an active item of the form $[(A, d) \rightarrow \bullet(B, d_1)\ \alpha,\ \psi]$, it will be combined with a passive item $[(B, d') \rightarrow \bullet\gamma,\ \phi]$, if (B, d_1) and (B, d') unify and their semantics do not overlap (as before Table 3.2) and if:

- (B, d_1) is of type `subst`, the intersection of the set *required indices* of the active item with the set *offered index* of the passive item is not empty.

- (B, d_1) is of type `adj`, the intersection of the set *offered index* of the active item with the set *required indices* of the passive item is not empty.

Blocking the proliferation of incomplete intermediate structures

In the FB-RTG derivation of the noun phrase *Une voiture rouge* followed in Section 3.1.3 we have choosen at each step the successful expansions. Our algorithm will systematically try all possible combinations and will therefore, produce intermediate incomplete structures such as *Une voiture* (cf. Section 2.1.1 where we discuss the lack of ordering information and intersective modification problem).

Following Kay (1996) and Carroll and Oepen's (2005) approach for blocking intermediate incomplete structures, we add a constraint on the composition operation. That is, when combining a passive item with an active one at a given substitution node X, it is required that the lower level derivation tree fragment (passive item) inserted into a higher level derivation tree structure (active item) covers all semantic predications that have a semantic index in common with the semantic index associated to node X.

To illustrate the working of this constraint let us take again the FB-RTG grammar in Section 3.1.3 p.54. We draw below a simplified version of the rules, i.e. only non-terminals and index features, that we will use for the example:

$$\textbf{r1.}\quad NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\\ \mathbf{b}\ [\text{idx}\quad \text{v}]\end{bmatrix} \rightarrow une(\ NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T}\\ \mathbf{b}\ [\text{idx}\quad \text{B}]\end{bmatrix}\ Det_A\ )$$

$$l_1 : une(v, h_r, h_s), qeq(h_r, l_2)$$

$$\textbf{r2.}\quad S_S \rightarrow achete(\ S_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T}\\ \mathbf{b}\ [\text{idx}\quad \text{O}]\end{bmatrix}\ NP_S\left[\mathbf{t}\ [\text{idx}\quad \text{t}]\right]\ VP_A\begin{bmatrix}\mathbf{t}\ [\text{idx}\quad \text{O}]\\ \mathbf{b}\ [\text{idx}\quad \text{E}]\end{bmatrix} V_A\begin{bmatrix}\mathbf{t}\ [\text{idx}\quad \text{E}]\\ \mathbf{b}\ [\text{idx}\quad \text{e}]\end{bmatrix}\ NP_S\left[\mathbf{t}\ [\text{idx}\quad \text{v}]\right])$$

$$l_5 : achete(e, t, v)$$

$$\textbf{r3.}\quad NP_S\left[\mathbf{t}\ \boxed{1}\right] \rightarrow voiture(\ NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T}\\ \mathbf{b}\ [\text{idx}\quad \text{v}]\end{bmatrix})$$

$$l_2 : voiture(v)$$

$$\textbf{r4.}\quad NP_A\left[\mathbf{t}\ \boxed{1}\right] \rightarrow rouge(\ NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{T}\\ \mathbf{b}\ [\text{idx}\quad \text{v}]\end{bmatrix}\ Adj_A\ )$$

$$l_2 : rouge(v)$$

$$\textbf{r6.}\quad NP_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{P}\\ \mathbf{b}\ \boxed{1}\ \text{P}\end{bmatrix} \rightarrow \epsilon \qquad \textbf{r7.}\quad S_A\begin{bmatrix}\mathbf{t}\ \boxed{1}\ \text{P}\\ \mathbf{b}\ \boxed{1}\ \text{P}\end{bmatrix} \rightarrow \epsilon$$

Let us also assume that the input semantics is that given in (26) for the sentence *Tex achète une voiture rouge*. At some point of the generation process for this sentence, the partial derivation given by the active item (3.1) will be produced. Following a sequence of bottom-up completions RTGen's algorithm will also produce the passive item in (3.2), corresponding to the noun phrase *une voiture rouge* and the one in (3.3), corresponding to the noun phrase *une voiture*. This last one is also licensed by the rules of the grammar and will be derived, nevertheless it does not contain the modifier *rouge*.

Both of the complete items, (3.2) and (3.3), could be used to rewrite the active symbol after the dot in the active item (3.1). And further two new passive items would be constructed. One resulting from the combination of (3.1) with (3.2) corresponding to the sentence *Tex achète une voiture rouge*. The other obtained from the combination of (3.1) with (3.3) corresponding to the sentence *Tex achète une voiture*. This last been an incomplete sentence because it does not cover the input semantics.

(26) $\{l_5 : achete(e,t,v),\ l_6 : tex(t),\ l_2 : rouge(v),\ l_2 : voiture(v)\ l_1 : une(v,h_r,h_s),\ qeq(h_r,l_2)\}$

$$[S_S \rightarrow achete(\ S_A\ NP_S\ VP_A\ V_A\ \bullet\ NP_S\ \mathbf{t}\ \begin{bmatrix}\text{idx} & \text{v}\end{bmatrix}\ ), \{l_5 : achete(e,t,v), l_6 : tex(t)\}] \tag{3.1}$$

$$[NP_S\begin{bmatrix}\mathbf{t}\ [\text{idx} & \text{v}]\\ \mathbf{b}\ [\text{idx} & \text{v}]\end{bmatrix} \rightarrow voiture(\ NP_A\ \mathbf{b}\ \begin{bmatrix}\text{idx} & \text{v}\end{bmatrix}\bullet\ ),\ \{l_2 : rouge(v),\ l_2 : voiture(v)\ l_1 : une(v,h_r,h_s), qeq(h_r,l_2)\}] \tag{3.2}$$

$$[NP_S\begin{bmatrix}\mathbf{t}\ [\text{idx} & \text{v}]\\ \mathbf{b}\ [\text{idx} & \text{v}]\end{bmatrix} \rightarrow voiture(\ NP_A\ \mathbf{b}\ \begin{bmatrix}\text{idx} & \text{v}\end{bmatrix}\bullet\ ),\ \{l_2 : voiture(v)\ l_1 : une(v,h_r,h_s), qeq(h_r,l_2)\}] \tag{3.3}$$

To block the construction of such incomplete sentences (i.e. 3.1 combined with 3.3). The blockin gconstraint verifies that when rewriting the non-terminal NP_S after the dot in 3.1 all predications in the input semantics containing the v index are covered by the passive item used to rewrite the non-terminal. As 3.3 does not cover $l_2 : rouge(v)$, it will not be combined with 3.1.

Unpacking the generation forest

The unpacking of the generation forest consists in extracting each individual derivation tree out of the derivation forest. To do this, the basic extraction procedure starts from each goal item of the form $[S' \rightarrow S_S\bullet, \phi]$ where ϕ is the input semantics (cf.

Figure 3.15: Derivation tree with an NP containing two pre-modifiers and a relative clause. Sentence *La gentille petite tatou qui dort chante* (The kind small armadillo that sleeps sings).

Section 3.2.1) in the chart, and systematically extracts all different possible derivation trees from the combination of all different possible derivations at each packing point.

In particular, different orderings of noun modifiers are packed into a single intermediate derivation structure. For instance, given the following NP with 3 intersective modifiers, *An easy introductory linguistic course* the different orderings (i.e. permutations) of the three modifiers, and in turn, their different derivations, will be packed into a single chart item. For the case of intersective modifiers packing, to extract only one derivation tree with the best modifiers orderings we could rely on a data-driven approach. For instance, Mitchell *et al.* (2011) propose an n-gram based model to obtain the best order for a given set of prenominal modifiers. This model was found to outperform other approaches, in a semi-supervised setting obtaining large amounts of data by using an automatic parser.

We integrated Mitchell *et al.*'s n-gram approach in our unpacking procedure. In this approach, the n-gram model is built on *extracted multiple modifiers NPs* –this gave better results than building the n-gram model based on entire sentences. An extracted NP (also called in [Mitchell *et al.*, 2011] as "simple NP") is a maximal NP that includes pre-modifiers such as determiners and adjectives but no post-nominal constituents such as prepositional phrases or relative clauses. When the unpacking procedure detects a packed NP item we build all alternative NP sub-phrases (cf. string extraction in next section) rank them and choose the one (and the corresponding sub-derivation tree) with highest score. One tricky point about integrating Mitchell *et al.*'s (2011) approach is how to "detect the simple NP". If we look at Figure 3.15, the sub-derivation that corresponds to the NP combines the pre and post modifiers. Then, we need to selectively extract certain constituents from the NP which form a simple NP and do the ranking based on these phrases.

3.2.3 String extraction from derivation trees

As Schmitz and Le Roux (2008) point out, the string language of TAG can also be encoded in a FB-RTG provided the FB-RTG of TAG derivations is extended with topological information along the lines of Kuhlmann (2007). This means that words could be extracted in the appropriate order without the need of explicitly building the phrase structure tree (i.e. derived tree). In what follows, we explain how we extend the encoding of Schmitz and Le Roux to support string extraction from the derivation tree.

Main points from Kuhlmann's (2007) framework

We will first give the intuitions behind the procedure which we follow to read-off the strings directly from derivation trees instead of building the corresponding derived tree. For more details and formal definitions we refer the reader to [Kuhlmann, 2007].

Kuhlmann (2007) draws on the observation that lexicalised grammars can be seen as generators of dependency trees. If the grammar is lexicalised then there is a one-to-one correspondence between the nodes in a derivation tree and the words (or positions) in the derived string, Figure 3.16 shows an example. Then, if the nodes of the derivation tree are ordered according to string positions of their anchors a dependency structure [19] is obtained. The dependency structure is said to be "induced" by the derivation.

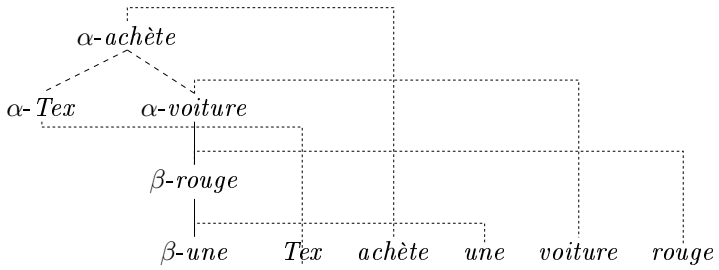


Figure 3.16: One-to-one correspondence between nodes in a derivation tree of a lexicalised grammar and words of the generated string.

In order to characterise grammatical formalisms in terms of the type of dependency structures that their derivations induce, Kuhlmann (2007) defines an algebraic framework. First, in this algebraic framework he shows: (i) how dependency structures can be encoded into terms over a certain signature of *order annotations* and (ii)

[19] A dependency structure for a sentence $\vec{w} = w_1.....w_n$ is the directed graph on the set of positions for $\vec{w}$ that contains an edge $i \rightarrow j$ if and only if w_j depends on w_i.

defines a *dependency algebra* where order annotations are interpreted as composition operations on dependency structures. Then, this framework is used to characterise the types of dependency structures produced by different grammatical formalisms. To formalise the link between grammar derivations and dependency structures, he starts by defining an algebra of the grammar derivations (*derivation algebra*). From this algebraic definition of the grammar, a string algebra and linearisation algebra are further defined. It follows from the linearisation algebra that productions (i.e. rules) of the grammar in a derivation can be seen as order annotations.

Here, we are not interested in the induced dependency structure per se, but in the linearisation of tree nodes. To introduce the main concepts of Kuhlmann's framework, we will start by reviewing the algebraic characterization of projective dependency structures and how context-free grammar derivations induce this type of dependencies.

In a dependency structure there are two relations, *governance*: the dependency relation between nodes, and *precedence*: a total order of the nodes of the graph. When imposing a global order (post or pre order) on the nodes in a tree, a dependency structure is obtained. The governance relation is the same but the nodes are ordered. Figure 3.17(a-b) shows a tree and a dependency structure induced by imposing an order (that of pre-order traversal of the tree) on tree nodes.

However, by giving the order following a tree traversal strategy such as pre-order, the position of a node with respect to its children is tied to the traversal strategy. There is a more flexible way to define order for the nodes of a tree that is not determined by the traversal strategy. It is to associate each node with its position relative to its children (if any) and make the traversal strategy use this information. For instance, in the tree in Figure 3.17(c) each node is associated with information (a list) of its order with respect to the order of its children. In this example, the tree node **1** is annotated with **[215]**, in terms of a traversal order this means that the first child node **2** is visited first, then the parent node himself, i.e. **1**, and finally the child node **5**. If this tree is traversed according to these order annotations, the nodes are linearised as shown in 3.17(d). The local tree formed by a node u and its children is called *treelet*, and a tree is a *treelet-ordered* tree if each of its nodes is annotated with a total order on the nodes in the treelet rooted at that node.

Order annotations. The list-based order annotations (e.g. tree in Figure 3.17(c) can be seen as a ranked set Ω and treelet-ordered trees as terms over this set T_Ω. The sequences in Ω are "node names" rather than concrete nodes. The node names are solved according to the term structure (i.e. their position, e.g. $1st$ child, $2nd$ child, etc.). Figure 3.18 shows the term for the treelet-ordered tree of Figure 3.17(c).

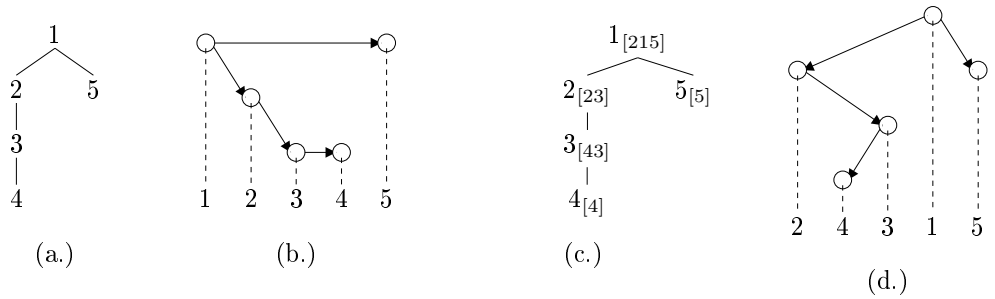


Figure 3.17: A children-ordered tree (a.) and the dependency structure induced by a pre-order traversal (b.) and a treelet-ordered tree (c.) and the dependency structure obtained by treelet-order traversal.

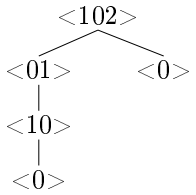


Figure 3.18: Term for the treelet-ordered tree of Figure 3.17c.

Up to here, we have informally described how projective dependency structures can be encoded as terms over order annotations (cf. [Kuhlmann, 2007] for the complete definition with the dependency algebra using the order annotations to build the dependency structure). Hereafter, we will see how an algebraic view on a grammatical formalism (we follow the case of context-free grammars) is set up to relate derivation structures of grammatical formalisms to dependency structures. Again, we are not interested in the resulting dependency structures but in how we associate these order annotations to the derivations of a grammar formalism.

So now, how to associate derivations with this kind of order annotations? The starting point is to take an algebraic view of the given grammatical formalism. For instance, in the case of a lexicalised CFG G the set of productions P of G can be turned into an algebra and the derivations can be seen as terms in this algebra. Further, a *string algebra* can be defined as having a string composition operation that takes as domain the set of terms in the derivation algebra. For instance, for each production $p = A \rightarrow A_1 \cdots A_{k-1} \cdot a \cdot A_k \cdots A_m$ of G with $A, A_1, ..., A_m$ non-terminals and a a terminal symbol and $\vec{a}$ strings of terminal symbols, the string composition operation is defined as:

$$f_p(\vec{a}_1, ..., \vec{a}_m) = \vec{a}_1 \cdots \vec{a}_{k-1} \cdot a \cdot \vec{a}_k \cdots \vec{a}_m$$

.

Each composition operation f_p concatenates the anchor (i.e. the terminal symbol a) of p and the strings obtained from the sub-derivations in the order specified by p.

Further, instead of evaluating derivations into strings of words (anchors of the productions in the derivation), it is possible to obtain a list of the nodes of the derivation, i.e. a linearization. For this, a *linearization algebra* is defined. To this end, for the case of a CFG G, a composition operation f_p is defined which concatenates node positions (e.g. gorn addresses) in the derivation structure. Then, for each production $p = A \rightarrow A_1 \cdots A_{k-1} \cdot a \cdot A_k \cdots A_m$ with $A, A_1, ..., A_m$ non-terminals and a a terminal symbol. A composition operation f_p is defined as follows, being $\vec{u}$ strings of node addresses (e.g. gorn addresses) and pfx_i a function that prefixes nodes with node addresses:

$$f_p(\vec{u}_1, ..., \vec{u}_m) = pfx_1(\vec{u}_1) \cdots pfx_{k-1}(\vec{u}_{k-1}) \cdot \epsilon \cdot pfx_k(\vec{u}_k) \cdots pfx_m(\vec{u}_m)$$

.

Each composition operation f_p of the linearization algebra concatenates a root node (i.e. the anchor of p) and appropriately prefixed (i.e. pfx applied) sub-derivations in the same order as they would be concatenated in the string algebra.

As G is lexicalised, the linearization algebra defines a bijection between the set of nodes in the derivation tree t and the set of positions in the derived string. Important to note is the fact that if we read the anchors of the productions in the derivation tree in the order specified by the linearization we obtain the derived string.

Then, the linearisation semantics of the derivations in the CFG mimics the treelet-ordered tree traversal. Hence, the productions of the grammar can be seen as order annotations and each derivation tree can be seen as a treelet-ordered tree. So, the translation between a derivation tree in a CFG and a projective dependency structure (represented as terms as described at the beginning of this section) can be obtained by *relabeling* the productions of G as order annotations as follows:

$$A \rightarrow A_1 \cdots A_{k-1} \cdot a \cdot A_k \cdots A_m \leftrightarrow \langle 1 \cdots (k-1) \cdot 0 \cdot k \cdots m \rangle$$

We have succinctly discussed the elements that we need from Kuhlmann's framework for the case of CFGs. We will now see how this framework applies to our

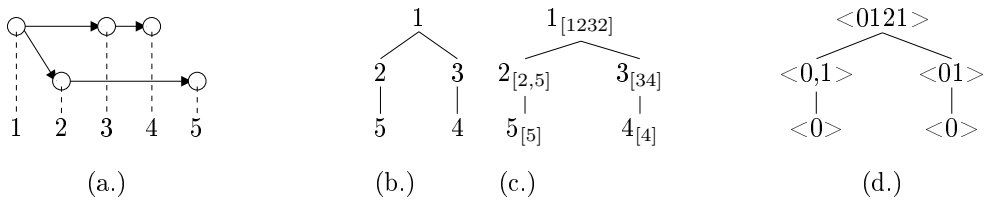


Figure 3.19: (a) Dependency structure, (b) tree, (c) block-ordered tree and (d) term.

grammar framework. That is, how we associate RTG rules with the kind of order annotations seen so far.

Extracting strings from TAG derivation trees

As is known from [Bodirsky *et al.*, 2005; Kuhlmann, 2007], TAGs produce well-nested dependencies with block-degree at most 2. We explain these two concepts with the example in Figure 3.19(a). The **yield** of the tree node **2** in the dependency structure in (a.) falls into two discontinuous spans, i.e. into two blocks, one is the block **[2]** and the other is **[5]**. Since this is the maximal number of blocks per yield, in the entire dependency structure D in (a.), it is said that the block degree of D is 2. The dependency structure in 3.19(a) is well-nested in that the edges $1 \rightarrow 2$ and $2 \rightarrow 5$ overlap but the node **2** dominates **5**.

Given the tree in Figure 3.19(b), it is possible to define a traversal order (o precedence relation) based on order annotations to order the nodes in the tree in such a way to induce the dependency in Figure 3.19(a). List-based order annotations can be defined in similar way as done in the previous section, the difference is the order annotations needed to account for discontinuity. The tree in Figure 3.19(b) can be annotated as shown in Figure 3.19(c). The tree traversal strategy works in a similar way as described for single list-based order annotations. We can follow the traversal example of tree 3.19(c) [20]. The root node **1** is labelled with the order [1232], this means that the yield of node **2** is distributed in two spans. So, we list node **1** then its daughter node **2**. The traversal of the sub-tree rooted at node **2**, is defined by the two-block list-based order annotation [2, 5]. The first component stipulates that the current node, i.e. **2**, should be visited (linearised). After processing the first component, the traversal continues at root node (i.e. [1232]) with the child node **3**. From this child, we list nodes **3** and **4**. Then, the order annotation follows by child node **2** (i.e. [2, 5]), and, at this time, the traversal continues with the second

[20] These trees are called block-ordered trees and they follow certain requirements on the lists annotating the nodes, see [Kuhlmann, 2007] p.40 for the complete definition.

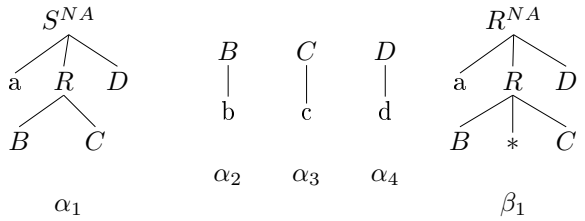


Figure 3.20: Toy TAG grammar

component of the order annotations, i.e. with node **5**. The result of this traversal gives the node linearisation 12345. If we replace node labels in the order annotations with node positions (i.e. $1st$, $2nd$ child and **0** for current node) we get the term annotations shown in Figure 3.19(d).

We have previously seen how the rules of CFG can be seen as defining order annotations. We will see now how TAG elementary trees can also define order annotations.

To analyse the string or linearization semantics of TAG elementary trees, we will follow an example using the toy grammar in Figure 3.20 [21]. In a TAG derived tree, we can read the yield string by reading the leaf (terminal) nodes from left-to-right. Similarly, to understand the string order in a TAG elementary tree we have to look at the nodes in its frontier from left-to-right. Let us look at the tree α_1 in Figure 3.20. Since no adjunction can take place at the root of α_1, the leftmost leaf in a possible derivation starting from α_1 is the leaf node labelled with the terminal symbol a. Then, the following leaf node is the node labelled with the non-terminal symbol B. However, before the string material that might come from B we have to consider that an adjunction might take place at the internal node R of α_1. More precisely, after the leaf node yielding the string a we expect the string material contributed by the left half (with respect to the foot node) of a possibly adjoined tree (eg. β_1). Then, all string material of the left half of an adjoined tree precedes the string material that is dominated by the adjunction site, i.e. B followed by C. All the material in the right half of the adjoined tree follows the content contributed by C. Finally, at the right-most leaf of α_1 labelled with non-terminal D we expect the string material contributed by D. Then, for α_1 we have the sequence $a\ R_1\ B\ C\ R_2\ D$ of string material.

On top of an appropriate algebraic formulation of TAG elementary trees (i.e. our grammar productions) we can define the string and linearization algebras. As Kuhlmann points out, from the linearization point of view TAGs correspond to Cou-

[21] The toy grammar example is taken from [Kuhlmann, 2007] p.85

pled Context-Free Grammars (CCFG) with rank at most 2 (CCFG(2), cf. [Kuhlmann, 2007] for the formal definition of CCFG). Then, we follow his definition of string algebra and linearization algebra for CCFG (which are in turn a special case of Linear Context-Free Rewriting Systems (LCFS, [Kallmeyer, 2013])). That is, we think of TAG elementary trees as CCFG rules.

Let us take the example CCFG grammar G from [Kuhlmann, 2007], with the alphabet of non-terminal symbols $\mathit{\Pi}_G = \{S/1, R/2, B/1, C/1, D/1\}$ (non-terminal symbols are ranked, the number after the slash represents the non-terminal's rank), an alphabet of terminal symbols $T_G = \{a, b, c, d\}$, and the start symbol of G is $S_G = S$, the set of productions is the following:

$$S \rightarrow \langle\ aR_1BCR_2D\rangle \mid \langle aBCD\rangle$$
$$R \rightarrow \langle\ aR_1B,\ CR_2D\rangle \mid \langle aB,\ CD\rangle$$
$$B \rightarrow \langle b\rangle, \quad C \rightarrow \langle c\rangle, \quad D \rightarrow \langle d\rangle$$

In this grammar, the right-hand side of the rules is a tuple of arity equal to the arity of the non-terminal in their left-hand side (e.g. R is of rank 2 therefore the right-hand side of productions rewriting R are tuples of arity 2). In addition, a non-terminal symbol appears in the right-hand side as many times as its arity to account for its synchronised rewriting (e.g. R appears in the right-hand side of the rules split into R_1 and R_2 meaning that they will be expanded at the same time). We can think of TAG initial trees as CCFG rules with non-terminals of rank 1 at the left-hand side, and auxiliary trees as rules with non-terminals of rank 2 in the left-hand side. The right hand-side of the rules would be derived from accommodating non-terminals of the TAG elementary tree according to the analysis of its string semantics. Internal nodes admitting adjunction are split into two non-terminals. For auxiliary trees (rules of rank 2), the right-hand side content is divided on the tree material to the left and to the right of the foot.

Thinking of elementary trees as productions of a CCFG(2) we can reuse the string and linearization algebras defined for CCFGs in [Kuhlmann, 2007]. But now, we directly go to the relabeling operation to obtain the order annotations.

Applying the relabeling function for the TAG trees α_1 and β_1 respectively from Figure 3.20 but reformulated as CCFG(2) rules as before:

$$relab(\alpha_1) = [012314]$$
$$relab(\beta_1) = [012,\ 314]$$
$$relab(\alpha_2) = [0]$$
$$relab(\alpha_3) = [0]$$
$$relab(\alpha_4) = [0]$$

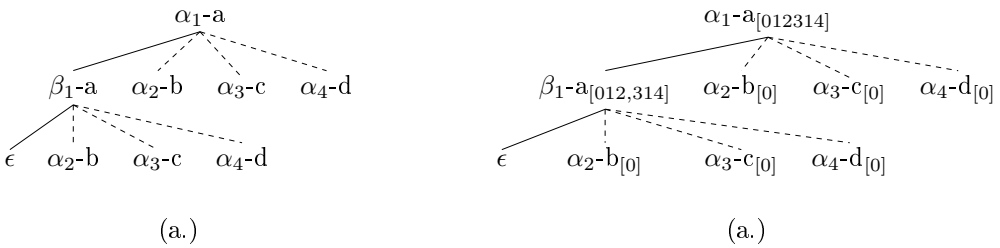


Figure 3.21: (a.) FB-RTG derivation tree and (b.) derivation tree with order annotations using the grammar in Figure 3.20 for the string *aabbccdd*.

Finally, these order annotations can be associated to each RTG rule $X \rightarrow \tau(X_1, \cdots, X_n)$ (in turn having been translated from a TAG elementary tree). Figure 3.21(a) show an example derivation tree for the string *aabbccdd* generated by the toy TAG grammar in 3.20. As now we can associate each node of the derivation tree with order annotations corresponding to each RTG rule, Figure 3.21(b) shows the same derivation tree but nodes are decorated with order annotations. If we traverse the derivation tree according to these order annotations, we can linearise its nodes and further read the derived string. For instance, the annotation [012314] states that first comes the current node anchor (i.e. a) then that of its first child (whose content is split in two, note 1 double appearing in the order annotations). This means that the traversal of the tree now continues with the content of its $1st$ child. On doing this, it follows the order annotations $[012, 314]$ (a 2-tuple) whose first component indicates that first comes the content of the current node (i.e. a), then the content of its $1st$ child and $2nd$ one. The first child is the empty string, the second is the terminal b. So far we have linearised the string *aab*. Before continuing with the second component of $[012, 314]$ the traversal returns to its parent (i.e. root node with annotations [012314]). It proceeds with the $2nd$ and $3rd$ children which contribute the strings *bc*, then the second component of the $1st$ child (second component of $[012, 314]$) and so on. At the end we have read the string *aabbccdd*.

3.3 Evaluation

To evaluate the performance of the RTGen algorithm, we use GENSEM ([Gottesman, 2009; Gardent *et al.*, 2010]) to construct two benchmarks. The first contains input cases involving intersective modifiers and the second contains input cases of varying overall complexity. Using these benchmarks, we examine the impact of the different optimisations incorporated in RTGen on its performance and we compare RTGen with an existing surface realiser, namely GENI ([Gardent and Kow, 2005]).

3.3.1 Surface realisers: GenI and RTGen configurations

GenI

GENI is a TAG-based surface realiser ([Gardent and Kow, 2005; Gardent and Kow, 2006; Kow, 2007]) which encodes a bottom up, tabular realisation algorithm optimised for TAGs. It implements the standard three step strategy (lexical selection, tree combining, sentence extraction) described in Section 3.1.2. GENI constructs derivation trees *bottom-up*. It starts from a set of selected lexical items and tries to combine them successively into larger structures (derived trees). The search is *depth-first* because the agenda it uses in its *chart-based* algorithm is implemented as a stack. Substitution nodes are processed in a (arbitrarily) fixed order (*left-to-right*) to avoid some spurious combinations stemming from different ways of processing nodes. Neither sub-tree sharing nor packing are implemented. Indexing is defined in terms of *semantic coverage* of chart items. Two major optimisations to deal with the complexity issues (cf. Section 2.1.1) are a filtering step and a two-phase tree combination (substitution and adjunction).

Polarity filtering. GENI's polarity filtering optimisation (based on [Bonfante *et al.*, 2004]) takes place between the lexical selection phase and the tree combination phase. The objective is to reduce the initial search space. As explained in Section 2.1.1, the number of combinations that are a priori possible after the lexical selection phase is $\prod_{1 \leq i \leq n} a_i$, with a_i the degree of **lexical ambiguity** of the i-th literal and n the number of literals in the input semantics.

The motivation for polarity filtering is based on the observation that not all the combinations of the selected lexical items would lead to a successful derivation. In specific, this filtering removes all tree sets covering the input semantics such that either the category of a substitution node cannot be canceled out by that of the root node of a different tree; or a root node fails to have a matching substitution site. In practice, as shown in Figure 3.22 each lexical item is assigned a polarity signature ($+Cat$ for each initial tree with root node category Cat; $-Cat$ for each substitution node with category Cat). Then, the total polarity signatures of each combination is computed [22]:

$\alpha_{d1} * \alpha_{tex} * \alpha_{corey} * \alpha_{e1} = 0$
$\alpha_{d1} * \alpha_{tex} * \alpha_{corey} * \alpha_{e2} = +1np, -1s$
$\alpha_{d2} * \alpha_{tex} * \alpha_{corey} * \alpha_{e1} = +1s, -1np$
$\alpha_{d2} * \alpha_{tex} * \alpha_{corey} * \alpha_{e2} = 0$

[22] An initial polarity of $-1s$ is considered in the computation of the polarity charge of a given combination.

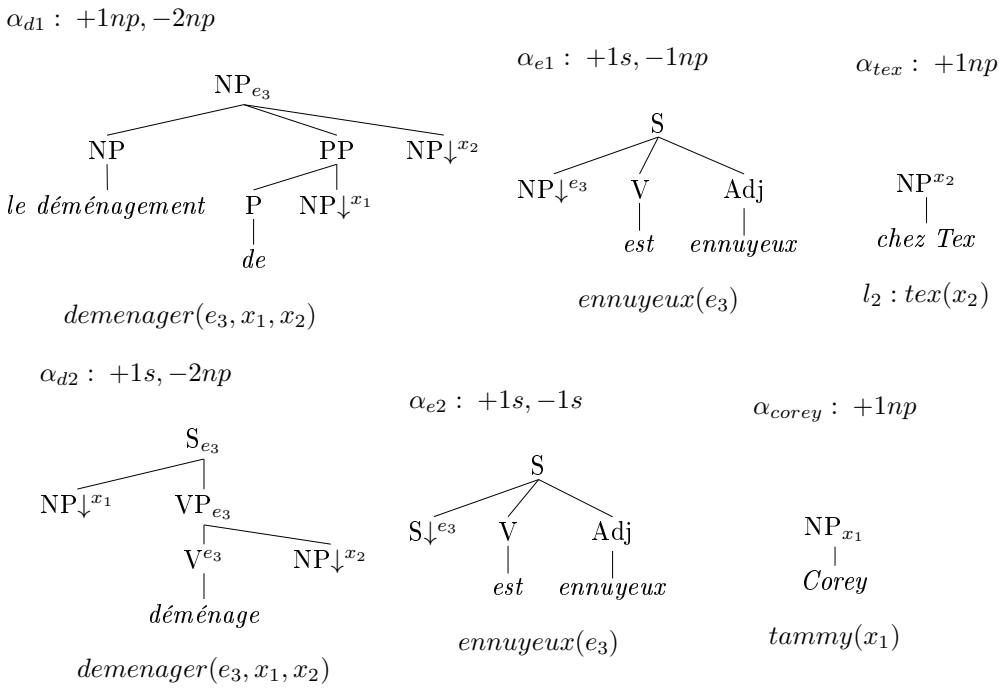


Figure 3.22: Selected lexical items with assigned polarities.

Only those combinations whose total polarity charge is equal to zero are considered during tree combination.

Polarity filtering has been shown to reduce surface realisation initial search space. However, this filtering relies solely on categorial information – feature information is not used. Furthermore, auxiliary trees have no impact on filtering since they provide and require the same category thereby being "polarity neutral elements".

Two-phase tree combination. Another optimisation technique is a delayed modification strategy inspired from [Carroll *et al.*, 1999]. Applying the composition operations (substitution and adjunction) in two separate phases supports the integration of a mechanism to deal with the intersective modifiers problem. In a first phase, only substitutions are carried out. In a second phase, modifiers are added by applying adjunction operations. In this way, the multiple modification structures licensed by modifications are produced but they are not multiplied out by the sentence context. This approach provides a mechanism to lessen the impact of intersective modifiers. Nevertheless, in the adjunction phase all possible permutations of modifiers are built, i.e. given n modifiers, $n!$ solutions are produced.

Different configurations of RTGen

FB-RTG and left-corner FB-RTG translation. The derivation tree generation algorithm of RTGen can be used with both the FB-RTG and the left-corner FB-RTG translations. Schmitz and Le Roux (2008) argue that the left corner translation should result in more predictive derivations. This observation might apply in particular to our Early-style algorithm, making more accurate top-down predictions and avoiding useless item compositions. In fact, in the left-corner mode epsilon unifications take place before and lhs might be more informative (i.e. containing instantiated features that might avoid items combinations)[23] (cf. Section 3.1.3).

Depending on how much linguistic information (i.e. feature constraints from the feature structures) is preserved in the FB-RTG rules, several RTGen configurations can be tried out which each reflect a different division of labor between constraint solving and structure building. To experiment with these several configurations, we exploit the fact that the FB-TAG-to-FB-RTG conversion procedure developed by Schmitz and Le Roux (2008) permits specifying which features should be preserved by the conversion.

RTGen-all. In this configuration, all the feature structure information present in the SemTAG elementary trees is carried over to the RTG rules. As a result, tree combining and constraint solving proceed simultaneously and the generated parse forest contains the derivation trees of all the output sentences.

RTGen-level0. In the RTGen-level0 configuration, only the syntactic category and the semantic features are preserved by the conversion. As a result, the grammar information used by the (derivation) tree building phase is comparable to that used by GenI filtering step. In both cases, the aim is to detect those sets of elementary trees which cover the input semantics and such that all syntactic requirements are satisfied while no syntactic resource is left out. A further step is additionally needed to produce only those trees which can be built from these tree sets when applying the constraints imposed by other features. In GenI, this additional step is carried out by the tree combining phase, in RTGen, it is realised by the extraction phase i.e., the phase that constructs the derived trees from the derivation trees produced by the tree combining phase.

RTGen-selective. Contrary to parsing, surface realisation only accesses the morphological lexicon last i.e., after sentence trees are built. Because throughout the

[23] Nevertheless, this depends on how the features are designed in the grammar.

tree combining phase, lemmas are handled rather than forms, much of the morpho-syntactic feature information which is necessary to block the construction of ill-formed constituents is simply not available. It is therefore meaningful to only include in the tree combining phase those features whose value is available at tree combining time. In a third experiment, we automatically identified those features from the observed feature structure unification failures during runs of the realisation algorithm. We then use only these features (in combination with the semantic features and with categorial information) during tree combining.

3.3.2 Constructing benchmarks for sentence generation

Unlike parsing, where the input (strings) can be taken from existing text, sentence generation requires abstract input data that is not readily available. Existing approaches to automated or semi-automated benchmark construction for sentence generation are of two main types depending on the type of sentence realiser used: either the sentence realiser is based on a reversible grammar and the benchmark items are constructed by parsing some sentences and selecting the appropriate semantic formula from the parser output; or it is not, and the benchmark items are derived by transformation from a syntactically annotated corpus.

To test a surface realiser based on a large reversible Head-driven Phrase Structure Grammar (HPSG), Carroll *et al.* (1999) use a small test set of two hand-constructed and 40 parsing-derived cases to test the impact of intersective modifiers on generation performance. Later on, Carroll and Oepen (2005) present a performance evaluation which uses as a benchmark the set of semantic representations produced by parsing 130 sentences from the Penn Treebank and manually selecting the correct semantic representations. Finally, White (2004) profiles a CCG[24]-based sentence realiser using two domain-focused reversible CCGs to produce two test suites of 549 and 276 ⟨ semantic formula, target sentence ⟩ pairs, respectively.

For realisers that are not based on a reversible grammar, there are approaches which derive large sets of realiser input from the Penn Treebank (PTB). For example, Langkilde-Geary (2002) proposes to translate the PTB annotations into a format accepted by her sentence generator Halogen. The output of this generator can then be automatically compared with the PTB sentence from which the corresponding input was derived. Similarly, Callaway (2003) builds an evaluation benchmark by transforming PTB trees into a format suitable for the KPML realiser he uses.

In all of the above cases, the data is derived from real world sentences, thereby

[24] Combinatory Categorial Grammar

exemplifying "real world complexity". If the corpus is large enough (as in the case of the PTB), the data can furthermore be expected to cover a broad range of syntactic phenomena. Moreover, the data, being derived from real world sentences, is not biased towards system-specific capabilities. Nonetheless, there are also limits to these approaches.

First, they fail to support graduated performance testing on constructs such as intersective modifiers or lexical ambiguity, which are known to be problematic for surface realisation.

Second, the construction of the benchmark is in both cases time consuming. In the reversible approach, for each input sentence, the correct interpretation must be manually selected from among the semantic formulae produced by the parser. As a side effect, the constructed benchmarks remain relatively small (825 in the case of White (2004); 130 in [Carroll and Oepen, 2005]). In the case of a benchmark derived by transformation from a syntactically annotated corpus, the implementation of the converter is both time-intensive and corpus-bound ([Callaway, 2003]). This coincides with recent results reported in [Belz *et al.*, 2011]: grammar based surface realisers faced a major obstacle in converting the shared task common ground input into the format "expected" by the system.

To avoid these shortcomings take a different approach. We make use of the GENSEM ([Gottesman, 2009; Gardent *et al.*, 2010]) tool to automatically generate focused benchmarks from the same FB-TAG grammar. In essence, GENSEM traverses the grammar to build semantic representations generated by this grammar. To ensure termination and linguistic coverage, user defined constraints are used (we refer the reader to [Gardent *et al.*, 2010; Gardent *et al.*, 2011a] for further details on GENSEM). With GENSEM, we can create tailored benchmarks that include specific constructions we want to test (e.g. those cases known problematic with lexical ambiguity and modifiers). Furthermore, we do not get into issues related to grammar or lexicon coverage while evaluating performance. In the next section, we describe two test suites for the evaluation of our surface realisation algorithm that we generate using GENSEM.

Two GENSEM benchmarks

We use GENSEM to produce benchmarks that are tailored to test the impact of optimizations to deal with (i) the intersective modifiers and (ii) lexical ambiguity issues.

The first benchmark (MODIFIERS) was designed to test the realisers on cases involving intersective modifiers. To support only this dimension, it displays little or

no variation w.r.t. other dimensions such as verb type and non-modifying adjuncts. It includes 1 789 input formulae with a varying number (from 0 to 4 modifications), type (N and VP modifications) and distribution of intersective modifiers (n modifiers distributed differently over the predicate argument structures). For instance, the formula in (27) involves 2 N and 1 VP modification. Further, it combines lexical ambiguity with modification complexities, i.e. for the *snore* modifier the grammar provides 10 trees.

(27) $l_1 : \exists(x_1, h_r, h_s), h_r \geq l_2, h_s \geq l_3, l_2 : man(x_1), l_2 : snoring(e_1, x_1), l_2 : big(x_1), l_3 : sleep(e_2, x_1), l_4 : soundly(e_2)$
(*A snoring big man sleeps soundly*)

The second benchmark (COMPLEXITY) was designed to test overall performance on cases of differing complexity (input formulae of increasing length, involving verbs with a various number and types of arguments and with a varying number of and types of modifiers). It contains 890 distinct cases. A sample formula extracted from this benchmark is shown in (28), which includes one modification and two different verb types.

(28) $h_1 \geq l_4, l_0 : want(e, h_1), l_1 : \exists(x_1, h_r, h_s), h_r \geq l_1, h_s \geq l_0, l_1 : man(x_1), l_1 : snoring(e_1, x_1), l_3 : \exists(x_2, h_p, h_w, h_u), h_p \geq l_3, h_w \geq l_4, h_u \geq l_5, l_3 : monkey(x_2), l_4 : eat(e_2, x_2, e_3)$
(*The snoring man wants the monkey to eat*)

3.3.3 Comparative results on GENSEM's benchmarks

To evaluate GENI and the various configurations of RTGEN base (RTGEN-all, RTGEN-level0, RTGEN-selective) with the left-corner transformed FB-TAG-to-FB-RTG translation, we ran the 4 algorithms in batch mode on the two benchmarks and collected the following data for each test case:

- Packed chart size : the number of chart items built. This feature is only applicable to RTGen as GENI does not implement packing.
- Unpacked chart size : the number of intermediate and final structures available after unpacking (or at the end of the tree combining process in the case of GENI). Note that RTGen never handles explicitly this number of intermediate structures. Even when carrying out unpacking as in that step it focuses only on successful structures.
- Initial Search Space (ISS) : the number of all possible combinations of elementary trees to be explored given the result of lexical selection on the input semantics. That is, the product of the number of FB-TAG elementary trees selected by each literal in the input semantics.

- Generation forest (GF) : the number of derivation trees covering the input semantics.

The graph in Figure 3.23 shows the differences between the different strategies with respect to the unpacked chart size metric.

A first observation is that RTGen-all outperforms GenI in terms of intermediate structures built . In other words, the Earley sharing and packing strategy is more effective in reducing the number of constituents built than the filtering and substitution-before-adjunction optimisations used by GenI. In fact, even when no feature information is used at all (RTGen-level0 plot), for more complex test cases, packing and sharing is more effective in reducing the chart size than filtering and operation ordering.

Another interesting observation is that RTGen-all and RTGen-selective have the same impact on chart size (their plots coincide). This is unsurprising since the features used by RTGen-selective have been selected based on their ability to block constituent combination. The features used in RTGen-selective mode are `wh`, `xp`, `assign-comp`, `mode`, `definite`, `inv`, `assign-case`, `rel-clause`, `extracted` and `phon`, in addition to the categorial and semantic information. In other words, using all 42 SemXTAG grammar features has the same impact on search space pruning as using only a small subset of them. As explained in the previous section, this is probably due to the fact that contrary to parsing, surface realisation only accesses the morphological lexicon after tree combining takes place. Another possibility is that the grammar is under constrained and that feature values are missing thereby inducing over-generation.

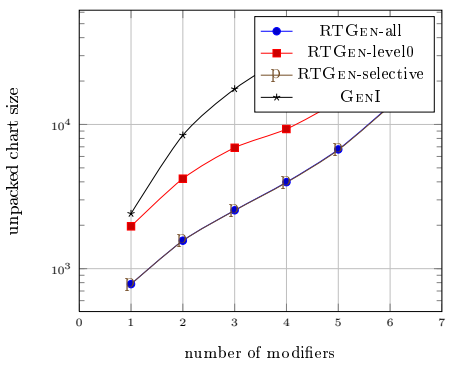


Figure 3.23: Performance of realisation approaches on the Modifiers benchmark, average unpacked chart size as a function of the number of modifiers.

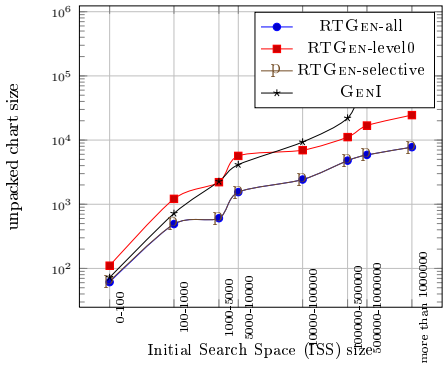


Figure 3.24: Performance of realisation approaches on the Complexity benchmark, average unpacked chart size as a function of the ISS complexity.

Zooming in on cases involving three modifiers, we show in Table 3.3 the average results for various efficiency metrics [25]. This provides a more detail view of the performance of the differences among the three RTGen variants.

strategy	GF	chart	unpacked-chart	seconds
RTGen-all				
RTGen-level0				
RTGen-selective				

Table 3.3: Average results on 610 test cases from the Modifiers benchmark. Each test case has 3 modifications, distributed in various ways between adjectival and adverbial modifications. The second column, Generation Forest (GF), is the number of derivation trees present in the generated parse forest. The third and fourth columns show the chart and unpacked chart sizes, respectively. The last column shows the runtime in seconds.

This data shows that running RTGen with no feature information leads not only to an increased chart size but also to runtimes that are higher in average than for full surface realisation i.e., realisation using the full grammar complete with constraints.

Interestingly, it also shows that the selective mode (RTGen-selective) permits improving runtimes while achieving almost perfect disambiguation in that the average number of derivation trees (GF) produced is close to that produced when using all features. The differences between the two generation forests stems from the fact that when some features are excluded some more derivation trees might be produced.

Graph 3.24 and Table 3.4 confirm the results obtained using the Modifiers benchmark on a testset (Complexity) where input complexity varies not only with respect to modification but also with respect to the length of the input and to the degree of lexical ambiguity. Typically, in a TAG, one word or one semantic literal may be associated either with one tree or with up to several hundred trees (e.g., ditransitive verbs and verbs with several subcategorisation types). By varying the type and the number of verbs selected by the semantic literals contained in the input semantics, the Complexity benchmark provides a more extensive way to test performance on cases of varying complexity.

We have chosen to use the left-corner mode in the previous evaluation because it provided more predictive derivations. Table 3.5 summarises in the upper part (RTGen base) information about the generation of the sentence *Tex looks for a master program that includes a module that includes a course*. It shows for both the FB-RTG and the left-corner FB-RTG translations the number of clashes encountered in pre-

[25] The two realisers being implemented in different programming languages (RTGen uses Prolog and GenI Haskell), runtimes comparisons are not necessarily very meaningful. Additionally, GenI does not provide time statistics. After adding this functionality to GenI, we found that overall GenI is faster on simple cases but slower on more complex ones. We are currently working on optimising RTGen prolog implementation before carrying out a full scale runtime comparison.

strategy	GF	chart	unpacked-chart	seconds
RTGen-all				
RTGen-level0				
RTGen-selective				

Table 3.4: Average results on 335 cases with $10000 < ISS \leq 100000$, from the COMPLEXITY benchmark. The columns show the same performance metrics as in Table 3.3.

diction and completion. As we can see the left-corner translation is more effective in blocking unwanted predictions. As a result, the number of items entered in the agenda is smaller (2473 against 1452), despite the fact that the number of rules in the left-corner FB-RTG is greater as auxiliary trees that adjoin into root positions are translated into two rules (cf. Section 3.1.3).[26] Note aside, by looking at the rows "Nb successful predictions" and "Nb items entered in agenda" we can see the effect of subsumption blocking of new edges and packing.

In the bottom part of Table 3.5, we summarise the same information but in this case obtained by running RTGen extended. After the optimisations the differences between the FB-RTG and left-corner FB-RTG disappear. Thus, the optimisations have a better impact. For instance, in the left-corner FB-RTG translation, instantiated semantic features are available in left-hand sides and therefore provide more information for predictions and completions. In RTGen extended, the indexing mechanism relies on the semantic indices of the semantic formula associated to each rule and blends this information with node type (substitution or adjunction site) information. Therefore, predications and combinations are effectively guided in both the FB-RTG and left-corner FB-RTG translations.

3.4 Related work on efficient surface realisation

3.4.1 Comparison with results in previous work

The results reported in previous work are not directly comparable with those obtained by other surface realisers because of differences in the size of the resources (grammars and lexicons); in the benchmarks used; in the programming language used for implementation and in the computers used for testing. Moreover the efficiency of existing surface realisers is often not reported on in the literature. In what follows, we compare our approach with two realisers for which such results were given namely, the HPSG based surface realiser described in [Carroll *et al.*, 1999] and the

[26] Note that "prediction and completion clashes" are illustrative in the sense that the counts are per prediction per feature. That is, in a failure to predict one rule, there might be more than one feature in conflict, therefore, summing more than one clash for each attempt of rule prediction.

	FB-RTG	*left-corner* FB-RTG
RTGen base		
Prediction clashes	437	9498
Completion clashes	4467	2675
Nb successful predictions	7490	6344
Nb items entered in agenda	2473	1452
RTGen extended		
Prediction clashes	287	6588
Completion clashes	2	109
Nb successful predictions	1887	2776
Nb items entered in agenda	429	623

Table 3.5: Summary of the number of predictions running the generation algorithms (Sections 3.2.1 and Section 3.2.2) for the generation of the sentence (and its licensed paraphrases) using the SemXTAG English grammar.

plan based approach of Koller and Hoffmann (2010). For this comparison, we run the extended RTGen-selective[27] with left-corner encoding instance of RTGen to generate the sentences in (29), (30), and (31). The first two sentences are used in [Carroll *et al.*, 1999] to illustrate the realiser efficiency. The third sentence, is added, to extend the comparison to a more complex case namely, a sentence including 4 verbs (main verb, sentential arguments and relative clauses) and 3 noun modifiers.

(29) *The manager in that office interviewed a new consultant from Germany.*

(30) *Our manager organised an unusual additional weekly departmental conference.*

(31) *Fido thinks John looks for a master program which includes a module which includes an easy introductory linguistics course.*

In Table 3.7, we show an excerpt of Carroll and Oepen's (2005) reported results. The results are broken down by average ambiguity rates, the first two columns show the average number of items and average sentence length in each partition. The other columns show relative CPU time. As explained in by Carroll and Oepen, the column $1p-f-$ corresponds to the baseline algorithm suggested by Kay (1996) implementing a one-phase without packing and without filtering algorithm. The subsequent column, headed $2p-f-$, corresponds to the algorithm proposed in [Carroll *et al.*, 1999] (two-phase processing of modifiers, no packing and no filtering). The last one, $1p+f+$, corresponds to the best-performing configuration reported in [Carroll and Oepen, 2005].

[27] Using the set of features: `assign-case, assign-comp, definite, extracted, idx, inv, mode, rel-clause rmode, wh, xp, rel-clause, phon, nocomp-mode, lemanchor`.

sentence	length	trees	init_search_space	chart	seconds
(29)	11	4	36	676	0.34
(30)	9	24	36	213	0.11
(31)	19	72	793152	989	1.06

Table 3.6: Summary of RTGen run on 3 sample sentences.

Aggregate	nb.items	length	1p-f-	2p-f-	1p+f+
$100 < trees \leq 500$	22	17.4	53.95	36.80	5.61
$50 < trees \leq 100$	21	18.1	51.53	13.12	3.74
$10 < trees \leq 50$	80	14.6	35.50	18.55	1.77
$0 < trees \leq 10$	185	10.5	9.62	6.83	0.58

Table 3.7: Extract of the results reported in Carroll and Oepen (2005).

In Table 3.6, we summarise RTGen results on the example sentences. The column *trees* shows the number of derivations produced while the column *init_search_space* corresponds to the number of potential combinations to be explored (cf. Section 2.1.1). There are 17 trees selected for *think* and 36 for *looks for* and *includes*. The last column gives the running time in seconds. We can think of sentences (29) to belong to the partition $0 < trees \leq 10$ of Carroll and Oepen's Table. Sentence (30) to be in the $10 < trees \leq 50$ partition and sentence (31) in $100 < trees \leq 500$. RTGen runtimes favourably compare to the results reported in [Carroll *et al.*, 1999] in the three cases considered.

Koller and Stone (2007) propose a planning-based approach which also constructs derivation trees rather than derived trees. As discuss in [Koller and Hoffmann, 2010], this planning approach fails to scale to conjunctions of five basic clauses such as *The man greets the man and the man greets the man and the man greets the man and the man greets the man and the man greets the man.* For such cases, all planners and planner strategies tries out by Koller and Hoffmann (2010) time out. In contrast, RTGen[28] yields the expected sentence in 2.03 seconds of CPU time. While the planning approach is an interesting way of dealing with the interactions between surface realisation and the generation of referring expressions, the grammar-based approach seems better suited to handle the production of well formed sentences of arbitrary length and complexity.

As discussed in Chapter 5, we plan to further evaluate RTGen in a more general setting, e.g. using Surface Realisation task data ([Belz *et al.*, 2011]). This would allow a more direct comparison with other surface realisers but require both obtaining efficiency results from these alternative realisers and more importantly, transforming

[28] RTGen-all, base algorithm and left-corner FB-RTG encoding.

the SR data into a format compatible with that expected by RTGen.

3.4.2 Encoding into another grammatical formalism

As already discussed in Section 3.1.1, the RTGEN approach is closely related to the work of Koller and Striegnitz (2002) where the XTAG grammar is converted to a dependency grammar capturing its derivation trees. This conversion enables the use of a constraint based dependency parser, a parser which was specifically developed for the efficient parsing of free word order languages and is shown to support an efficient handling of both lexical and modifier attachment ambiguity.

Our proposal differs from this approach in three main ways. First, contrary to XTAG, SEMXTAG integrates a full-fledged, unification-based compositional semantics thereby allowing for a principled coupling between semantic representations and natural language expressions. Second, the grammar conversion and the feature-based RTGs used by RTGEN accurately translates the full range of unification mechanisms employed in FB-TAG whereas the conversion described by Koller and Striegnitz (2002) does not take into account feature structure information. Third, the RTGEN approach was extensively tested on a large benchmark using 3 different configurations whilst Koller and Striegnitz's results are restricted to a few hand constructed example inputs.

3.4.3 Chart generation algorithm optimisations

Carroll and Oepen (2005) provide an extensive and detailed study of how various techniques used to optimise parsing and surface realisation impact the efficiency of a surface realiser based on a large coverage Head-Driven Phrase Structure grammar.

Because they use different grammars, grammar formalisms and different benchmarks, it is difficult to compare the RTGEN and the HPSG approach. However, one point is put forward by Carroll and Oepen (2005) which it would be interesting to integrate in RTGEN. Carroll and Oepen show that for packing to be efficient, it is important that equivalence be checked through subsumption, not through equality. RTGEN also implements a packing mechanism with subsumption check, i.e. different ways of covering the same subset of the input semantics are grouped together and represented in the chart by the most general one. One difference however it that RTGEN will pack analyses together as long as the new ones are more specific cases. It will not go backwards to recalculate the packing made so far if a more general item is found ([Oepen and Carroll, 2000]). In this case the algorithm will pack them under two different groups.

3.4.4 Statistical pruning

Various probabilistic techniques have been proposed in surface realisation to improve e.g., lexical selection, the handling of intersective modifiers or ranking. For instance, Bangalore and Rambow (2000a) use a tree model to produce a single most probable lexical selection while in White's system, the best paraphrase is determined on the basis of n-gram scores. Further, to address the fact that there are $n!$ ways to combine any n modifiers with a single constituent, White (2004) proposes to use a language model to prune the chart of identical edges representing different modifier permutations, e.g., to choose between *fierce black cat* and *black fierce cat*. Similarly, Bangalore and Rambow (2000a) assume a single derivation tree that encodes a word lattice (*a {fierce black, black fierce} cat*), and uses statistical knowledge to select the best linearisation. Our approach differs from these approaches in that neither lexical selection is filtered nor ranking is performed. However, it shares some similarities with respect to intersective modifiers handling. Our approach handles only one instance of the $n!$ ordering possibilities thanks to the chart packing strategy during generation, and extracts only one ordering if the prenominal modification ordering model is used during the unpacking phase.

3.5 Conclusions and perspectives

In this chapter we presented RTGen, a new surface realisation algorithm for TAG based on an RTG encoding of its derivation trees. This encoding facilitated the implementation of an Earley-style chart algorithm with sharing and packing of intermediate structures. It has also permitted varying the generation algorithm according to (i) the two different encodings, the original RTG translation and the left-corner transformed one, and (ii) the selective translation of features from the feature structures. In the implementation of the generator, we have integrated well-known techniques such as chart indexing, control of proliferation of intermediate incomplete structures and packing. Nonetheless, RTGen algorithm could be further extended.

In the start. Reducing the initial search space. The sole mechanism of RTGen to deal with lexical ambiguity is packing. However, known strategies to tackle this issue could be also integrated in RTGen. One of these strategies is lexical selection filtering which has been shown to be drastically effective in parsing ([Bangalore and Joshi, 1999; Chen *et al.*, 1999; Gardent *et al.*, 2011b]) as well as in generation ([Gardent and Kow, 2005; Espinosa *et al.*, 2008; Bangalore and Rambow, 2000a]). This filtering step might be based on symbolic information contained in the grammar and the input se-

mantics (e.g. *polarity filtering* in GenI [Gardent and Kow, 2005]), or data-driven (e.g. *hypertagging* in OpenCCG [Espinosa *et al.*, 2008] using an extended corpus of CCG derivations, logical forms and sentences derived from the CCGBank [Hockenmaier and Steedman, 2007] or *hierarchical supertagging* in FERGUS [Bangalore and Rambow, 2000a; Bangalore and Rambow, 2000b]). Filtering the initial search space reduces the amount of work carried out by the generation algorithm. Nevertheless, it is important to note that on top of it we need to integrate additional optimisation strategies (e.g. packing). The polarity filtering approach is cautious in that only infelicitous trees are pruned, however, as shown in our evaluation in Section 3.3.3, much work might still remain for the chart generator. Reinforcing the same fact, with the statistical filtering there is a trade-off between using a few best lexically selected items at the expense of (possibly) loosing coverage and using a bigger set of lexical selected items augmenting the cost of generation. Moreover, these models require the creation of an annotated corpus relating input semantic formulae and parse trees. The standard way to create this corpus would be to parse with a SemTAG-based parser. However, it would be interesting to explore how the GENSEM tool ([Gardent *et al.*, 2010; Gardent and Kruszewski, 2012]) could be used for creating such a corpus ([Hwa, 2000]).

In the forest. In the past decade, purely symbolic sentence generation algorithms have evolved through the incorporation of data-driven techniques into hybrid symbolic-statistical generation algorithms ([Carroll *et al.*, 1999; Carroll and Oepen, 2005; White, 2004; Espinosa *et al.*, 2008; Nakanishi *et al.*, 2005]). A commonality among these approaches is that all of them rely on a packed shared forest, in addition to statistical models to prune during or at the end of the generation process. RTGen provides an algorithm for TAG which builds a packed shared forest. Thus, for example, in a similar way as we choose the order of pre-modifiers, we could implement a selective unpacking mechanism based on a language model (e.g. [Langkilde, 2000]).

RTGen packing criteria could be relaxed in order to obtain more compact forests. For instance, by not considering some features as well as some right-hand side non-terminals of RTG rules packing of active edges could be favoured. Going further, we could take the following criteria for packing passive edges: consider as equivalent edges with the same semantic coverage though possibly different root (or left-hand side category), similarly to [Bangalore and Rambow, 2000a] γ-trees which do not specify exactly how they are inserted into another tree. For instance, in this way edges for S: *Tammy adore la tatou*(Tammy loves the armadillo) and NP: *La tatou que Tammy adore*(The armadillo that Tammy loves) would be packed into a [NP|S]

passive edge.

In conclusion, one interesting feature of hybrid approaches to NLG is that naturally support a distinction between items (or combinations thereof) which can be safely ignored because they cannot occur in a valid phrase structure tree (e.g. using polarity filtering) and those which should be preferred base on statistical information (e.g. using n-grams). These approaches have being applied to NLG for example by White (2004) and Rambow and Joshi (1994). For future work, it would be interesting to explore which approaches could be used to further extend RTGen.

Chapter 4

Natural language generation for language learning

Contents

Grammar exercises in textbooks are not built from arbitrary text material, the design of textbooks for language learning is informed by evidence from language learning and acquisition research and teaching experience ([Krashen, 1982; Pienemann, 1998; Biber and Conrad, 2010]). In general, their syntax and vocabulary are tailored to the lesson topic and in accordance with a given language level. Usually, these

exercises are edited by hand. Even though there exist authoring tools such as *Hot Potatoes*[29] ([Winke and MacGregor, 2001]) (cf. Chapter 2.3) that provide a graphical interface where the teacher can create different types of exercises, ultimately, the exercises are created by hand. The language teacher manually enters the text that serves as the stem for the exercise and further edits it, for instance by selecting blanks in a Fill-in-the-blank exercise type. In addition, the teacher must provide the solutions and the associated feedback messages. Another example along these lines is the system *Lämpel* developed within the context of the Allegro project[30]. In the final project evaluation report, the authors of Lämpel indicate that around 100 exercises of different types have been edited by language teachers since the platform was made available in 2011. In contrast, the approach we propose for automating exercise edition was shown to automatically provide around 5000 exercises from 50 input semantic formulae.

Our *SemTAG-based natural language generation* approach aims at automatically producing syntactically and lexically controlled grammar exercises for language learning. A key feature of this approach is that because the grammar constitutes a rich linguistic resource describing natural language, the sentences produced by the generation processed are automatically associated with detailed linguistic information. This permits specifying fine-grained linguistic constraints for selecting adequate exercice stems and supports further processing for building grammar exercises. Another important aspect of the approach is the input underspecification mechanism endorsed by our generator together with the paraphrastic power and coverage of the grammar. This alleviates content edition since from the underspecified input, we can obtain several syntactically and morpho-syntactically distinct output sentences.

In this chapter, we describe our approach for automatically generating grammar exercises. We start by describing the task of grammar exercises generation and introduce the framework we propose, namely *GramEx* (Section 4.1). In Section 4.2, we describe how we obtain appropriate exercise stems for building grammar exercises using a constraint language over the syntactic and morpho-syntactic properties of the *SemTAG* grammar and retrieving those generated sentences that satisfy the constraints modelling a given exercise. In Section 4.3, we show how to generate Fill-In-the-Blank (FIB) and Shuffle exercise types. In Section 4.4, we move on to the generation of transformation-based type of exercises and present an approach for generating pairs of sentences that are related by a syntactic transformation. We evaluate the framework on several dimensions using quantitative and qualitative

[29] `http://hotpot.uvic.ca/`
[30] `http://www.allegro-project.eu/`

metrics as well as a small scale user-based evaluation. Finally, we wrap up with a discussion of conclusions and perspectives (Section 4.6). Throughout this chapter, our reference grammar book is the *Tex's French Grammar* online grammar book. Pedagogical goals as well as most of the examples are taken from it.

4.1 Introduction

To generate grammar exercises we need some "source" text material (i.e. sentences or phrases) from which concrete exercises can be built. We call this text material *exercise stem*. Grammar exercises pursue different pedagogical goals; and these pedagogical goals impose syntactic constraints on the stems used to build these exercises. In practice, exercise stems are constrained by the:

exercise stem

stems for specific pedagogical goals

Goal linguistic phenomena. Sentences for exercising on a given pedagogical goal should encompass the specific linguistic phenomena pursued by that pedagogical goal. For instance, given the pedagogical goal of learning *relative clauses*, a sentence used to build a grammar exercise satisfying that goal might contain a relative pronoun, e.g. *dont* in (32).

(32) *Tex: Le livre* ***dont*** *je suis l'auteur est un roman historique.*
*T*ex: The book of which I'm the author is an historical novel.

Preferred linguistic phenomena. The content and the ordering of the content in grammar books is carefully chosen by grammar books writers ([Krashen, 1982; Pienemann, 1998; Biber and Conrad, 2010]). Certain grammar structures are preferred to be introduced in earlier stages than others. Therefore, additional constraints might be placed on the general syntactic configuration of the selected sentences. Let us consider the case of an introductory lesson where the pedagogical goal is learning *adjectives*. In this case, sentences containing "simpler" grammatical constructions, such as (33) might be preferred. In contrast, more complex sentences (e.g. example (32)) with passive constructions and relative clauses, among others, might be avoided.

(33) *Tammy a une robe* ***ravissante****.*
*T*ammy has a ravishing dress.

Granularity of the pedagogical goal. A pedagogical goal of learning *adjectives* could be pursued in which sentences containing any adjective type might be used as

learning material. However, more fine grained pedagogical goals could be defined; for instance, the pedagogical goal of learning *irregular adjectives*. In this case, stem sentences for exercises on this pedagogical goal should include those adjectives that are of type irregular.

The first milestone of our approach is the generation of exercise stems that satisfy these pedagogical constraints. The next step, is the construction of concrete *exercise items* . That is, from generated stems we want to produce both the exercise *question* (Q) and the expected *solution* (S) –as well as alternative correct solutions in some cases. In (34), we can see two examples of question and solution, one (34a) for a Fill-in-the-blank and the other (34b) for a transformation type of exercise.

exercise item: question and solution

(34) a. Give the correct form of the adjective indicated in parentheses.
Q: Tammy a une voix _____. (doux, 'soft') (Tammy has a soft voice.)
S: douce

b. Rewrite each question using the form specified in parentheses, est-ce que or n'est-ce pas.
Q: Tex aime Bette? (est-ce que) (Tex loves Bette?)
S: Est-ce que Tex aime Bette?

GramEx

In the following sections, we describe the *GramEx* framework for the generation of grammar exercises. That is, we will see in Section 4.2 how exercise stems are produced by *GramEx*; and how these stems are further processed to form the question (Q) and solution (S) of Fill-in-the-blank, Shuffle, and Reformulation exercises (Sections 4.3 and 4.4).

4.2 Generating exercise stems

To generate exercise stems, *GramEx* proceeds as follows. First, a *generation bank* is constructed using surface realisation techniques. This generation bank stores sentences that have been generated together with the detailed linguistic information associated by the generation algorithm with each of these sentences. Next, stem sentences that permit exercising the given pedagogical goal are retrieved from the generation bank using a *constraint language*. This language permits defining pedagogical goals in terms of the linguistic properties associated to the generated sentences (cf. Section 2.2.5).

4.2.1 Constructing a Generation bank

Sentence (S_i):
Tammy a une voix douce

Semantics (σ_i):
L0:proper_q(J HR HS) L1:named(J tammy_n)
L1:indiv(J f sg) qeq(HR L1) A:un_d(C RH SH)
B:indiv(C f sg) qeq(RH B) B:doux_adj(S C) B:voix_n(C)
G:avoir_v(E J C) G:event(E pres indet ind)

Morpho-syntactic and syntactic properties (L_i):

```
{lemma: "Tammy",
 lemma-features: {anim:+,num:sg,det: +,wh:-,cat:n,func:subj,xp: +,gen:f},
 synProperties: {propername}},
{lemma: "avoir",
 lemma-features: {aux-refl:-,inv:-,pers:3,pron:-,num:sg,mode:ind,aspect:indet,
                  tense:pres,stemchange:-,flexion:irreg},
 synProperties: {CanObj,CanSubj, Active, n0Vn1}},
{lemma: "un",
 lemma-features: {wh:-,num:sg,mass:-,cat:d,gen:f,def:+},
 synProperties: {determiner}},
{lemma: "voix",
 lemma-features: {func:obj,wh:-,cat:n,num:sg,mass:-,gen:f,flexion:irreg},
 synProperties: {noun}},
{lemma: "doux",
 lemma-features: {num:sg,gen:f,flexion:irreg,cat:adj},
 synProperties: {Epith,EpithPost}}
```

Derivation Tree (τ_i):

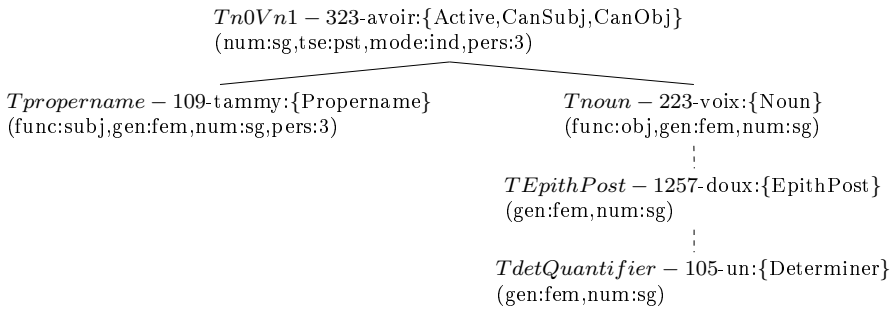


Figure 4.1: Linguistic information associated by *GraDe* with the sentence *Tammy a un voix douce* (Tammy has a soft voice).

generation bank

The generation bank is a collection of sentences associated with a representation of their semantic content and a detailed description of their syntactic and morpho-syntactic properties. In other words, a generation bank is a set of $(S_i, L_i, \sigma_i, \tau_i)$ tuples where S_i is a sentence, L_i is a set of linguistic properties true of that sentence, σ_i is its semantic representation, and τ_i is its associated derivation tree.

Figure 4.1 shows the linguistic information associated with the sentence S_i, *Tammy a une voix douce* (Tammy has a soft voice), output by our generator. The semantic representation of the sentence is given by σ_i, a MRS (Minimal Recursion

Semantics, [Copestake *et al.*, 2001]) representation. Each lexical item in the derived sentence is associated with a set of morpho-syntactic and syntactic properties L_i. That is, a set of feature-value pairs occurring as values of the `lemma-features` fields and a set of syntactic properties, i.e. items occurring in the `synProperties` fields. The last component is the *SemTAG* derivation tree τ_i which provides structural information. As can be seen at the bottom of Figure 4.1, tree nodes are labelled with (i) the name of an elementary tree, (ii) the set of tree properties associated with the elementary tree, (iii) the lexical item anchor of the tree, and (iv) a subset of the linguistic properties from L_i associated to the lexical anchor. Importantly, the derivation tree provides information about relations between lexical items, we will see more details about this in Section 4.4.2.

To produce those tuples, we use the *GraDe* grammar traversal algorithm described in [Gardent and Kruszewski, 2012]. Given a grammar G and a set of user-defined constraints, this algorithm generates sentences licensed by G. The user-defined constraints are either parameters designed to constrain the search space and guarantee termination (e.g., upper-bound on the number and type of recursive rules used or upper-bound on the depth of the tree built by *GraDe*); or linguistic parameters which permit constraining the output (e.g., by specifying a *core semantics* the output must verbalise or by requiring the main verb to be of a certain type). Note, that we mean by core semantics a set of literals describing predicate/argument and modifier/modifiee relationships and not a full semantic specification as the one used in the grammar (cf. Section 2.2.4). Indeed, the job of *GraDe* is to complete this input core semantics turning it into a well-formed full semantic representation. Here, we use *GraDe* both to generate from manually specified semantic input; and from a grammar (in this case an existing grammar is used and no manual input need to be specified). As explained in [Gardent and Kruszewski, 2012], when generating from a semantic representation, the output sentences are constrained to verbalise that semantics but the input semantics may be underspecified thereby allowing for morpho-syntactic, syntactic and temporal variants to be produced from a single semantics. For instance, given the input semantics *L1:named(J bette_ n) A:le_ d(C RH SH) B:bijou_ n(C) G:aimer_ v(E J C)*, *GraDe* will output among others the following variants:

core semantics

> Bette aime le bijou *(Bette likes the jewel)*, Bette aime les bijoux *(Bette likes the jewels)*, C'est Bette qui aime le bijou *(It is Bette who likes the jewel)*, C'est Bette qui aime les bijoux *(It is Bette who likes the jewel)*), Bette aimait le bijou *(Bette liked the jewel)*, Bette aimait les bijoux *(Bette liked the jewels)*, ...

When generating from the grammar, the output is even less constrained since all

derivations compatible with the user-defined constraints will be produced irrespective of semantic content. For instance, when setting *GraDe* with constraints restricting the grammar traversal to only derive basic clauses containing an intransitive verb, the output sentences include among others the following sentences:

> Elle chante (*She sings*), La tatou chante-t'elle? (*Does the armadillo sing?*), La tatou chante (*The armadillo sings*), Chacun chante -t'il (*Does everyone sing?*), Chacun chante (*Everyone sings*), Quand chante la tatou? (*When does the armadillo sing?*) Quand chante quel tatou? (*When does which armadillo sing?*), Quand chante Tammy? (*When does Tammy sing?*), Chante-t'elle? (*Does she sing?*) Chante -t'il? (*Does he sing?*), Chante! (*Sing!*), Quel tatou chante ? (*Which armadillo sings?*), Tammy chante-t'elle? (*Does Tammy sing?*), Tammy chante (*Tammy sings*), une tatou qui chante chante (*An armadillo which sings sings*), C'est une tatou qui chante (*It is an armadillo which sings*) , ...

4.2.2 Retrieving Appropriate Sentences

To enable the retrieval of sentences that are appropriate for a given pedagogical goal, we define a constraint language on the linguistic properties assigned by *GraDe* to sentences. We then express pedagogical goals constraints in that language; and we use the resulting specifications as queries to retrieve from the generation bank appropriate stem sentences. For instance, to retrieve a sentence for building a FIB exercise where the blank is a relative pronoun, we query the generation bank with the constraint *RelativePronoun*. This will return all sentences in the generation bank whose `synProperties` field contains the *RelativePronoun* item i.e., all sentences containing a relative pronoun.

GramEx Query Language

We now define the query language used to retrieve sentences that are appropriate to build an exercise for a given pedagogical goal. Let B be a generation bank and let $(S_i, L_i, \sigma_i, \tau_i)$ be the tuples stored in B. Then, a *GramEx* query q permits retrieving from B the set of sentences $S_i \in (S_i, L_i, \sigma_i, \tau_i)$ such that L_i satisfies q. In other words, *GramEx* queries permit retrieving from the generation bank all sentences whose linguistic properties satisfy those queries.

The syntax of the *GramEx* query language is as follows:

Grammatical Properties (traceItem)	
Argument Realisation	Cleft, CleftSUbj, CleftOBJ, ...,
	InvertedSubj
	Questioned, QuSubj, ...
	Relativised, RelSubj ...
	Pronominalised, ProSubj, ...
Voice	Active, Passive, Reflexive
Aux	tse, modal, causal
Adjective	Predicative,Pre/Post nominal
Adverb	Sentential, Verbal
Morpho-Syntactic Properties (feature=value)	
Tense	present,future,past
Number	mass, count, plural, singular
Inflexion	reg,irreg

Table 4.1: Some grammatical and morpho-syntactic properties that can be used to specify pedagogical goals.

LingDescription → LingTerm
LingTerm → LingFactor | LingTerm ∨ LingFactor
LingFactor → LingUnary | LingFactor ∧ LingUnary
LingUnary → LingPrimary | ¬ LingPrimary
LingPrimary → PrimitiveCond | **(** LingDescription **)** | **[** LingDescription **]**
PrimitiveCond → *traceItem* | $feature = value$

boolean constraints on linguistic properties

In words: the *GramEx* query language permits defining queries that are arbitrary boolean constraints on the linguistic properties associated by *GraDe* with each generated sentence. In addition, complex constraints can be named and reused (macros); and expressions can be required to hold of a single lexical item ([LingDescription] indicates that LingDescription should be satisfied by the linguistic properties of a single lexical item).

The signature of the language is the set of grammatical (*traceItem*) and morpho-syntactic properties ($feature = value$) associated by *GraDe* with each generated sentence where *traceItem* is any item occurring in the value of a `trace` field and $feature = value$ any feature/value pair occurring in the value of a `lemma-features` field (cf. Figure 4.1). Table 4.1 shows some of the constraints that can be used to express pedagogical goals in the *GramEx* query language.

Query Examples

The *GramEx* query language allows for very specific constraints to be expressed, thereby providing a fine-grained control over the type of sentences and therefore over the types of exercises that can be produced. The following example queries illustrate this.

(35) a. EpithAnte
Tex pense que Tammy est une jolie tatou
Tex thinks that Tammy is a pretty armadillo

b. [Epith ∧ flexion: irreg]
Tex et Tammy ont une voix douce
Tex and Tammy have a soft voice

c. POBJinf ∧ CLAUSE
POBJinf ≡ (DE-OBJinf ∨ A-OBJinf)
CLAUSE ≡ Vfin ∧ ¬Mod ∧ ¬CCoord ∧ ¬Sub
Tammy refuse de chanter
Tammy refuses to sing

Query (35a) shows a query for retrieving sentences containing prenominal adjectives which uses the grammatical (*traceItem*) property EpithAnte associated with preposed adjectives.

In contrast, Query (35b) uses both grammatical and morpho-syntactic properties to retrieve sentences containing a prenominal or postnominal adjective[31] with irregular inflexion. The square brackets in the query force the conjunctive constraint to be satisfied by a single lexical unit. That is, the query will be satisfied by sentences containing a lexical item that is both a pre or postnominal adjective and has irregular inflexion. This excludes sentences including, for instance, a postnominal adjective and a verb with irregular inflexion (unless they also contain an irregular adjective).

Finally, Query (35c) shows a more complex case where the pedagogical goal is defined in terms of predefined macros themselves defined as *GramEx* query expressions. The pedagogical goal is defined as a query which retrieves basic clauses (CLAUSE) containing a prepositional infinitival object (POBJinf). A sentence containing a prepositional infinitival object is in turn defined (second line) as a prepositional object introduced either by the *de* or the *à* preposition. And a basic clause (3rd line) is defined as a sentence containing a finite verb and excluding modifiers, clausal or verb phrase coordination (CCORD) and subordinated clauses. The expressions CCoord and Sub are themselves defined rather than primitive expressions.

4.2.3 GramEx implementation details and resources

The major three steps of our exercise generation approach are shown schematically in Figure 4.2.

[31] Note that in the grammar Epith ≡ (EpithAnte ∨ EpithPost)

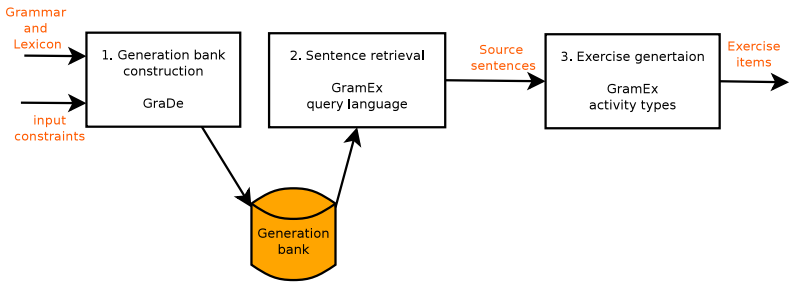


Figure 4.2: *GramEx* architecture.

The first module corresponds to *GraDe* (described in Section 4.2.1). It takes as input a grammar and a lexicon together with a set of traversal constraints to generate sentences. *GraDe* is written in Python[32] and converts the FB-RTG derived from the *SemTAG* Grammar to a Definite Clause Grammar which supports a constraint-driven traversal of the grammar. The output of *GraDe*, i.e. a sentence with its associated linguistic information as shown in Figure 4.1, is encoded in JSON (JavaScript Object Notation[33]) format and is stored in an open source document database named MongoDB[34].

The *GramEx*'s modules are written in Java. In the second step, the second module in Figure 4.2, *GramEx*'s boolean query expressions are translated into MongoDB queries to retrieve from the sentence bank those sentences that satisfy the constraints defined by the queries. The retrieved sentences are then passed on to the third module. In the last step, the third module implements the different exercise types by post-processing the selected exercise stems so as to produce exercise items. In the next section, we describe the exercise types defined in *GramEx*.

grammar and lexicons

The grammar used is SemFraG (an FB-TAG for French, cf. Section 2.2.5). This grammar contains around 1300 elementary trees and covers auxiliaries, copula, raising and small clause constructions, relative clauses, infinitives, gerunds, passives, adjuncts, wh-clefts, PRO constructions, imperatives and 15 distinct subcategorisation frames.

The syntactic and morpho-syntactic lexicons used for generation were derived from various existing lexicons, converted to fit the format expected by *GraDe* and tailored to cover basic vocabulary as defined by the lexicon used in *Tex's French Grammar*. The syntactic lexicon contains 690 lemmas and the morphological lexicon 5294 forms.

[32] www.python.org/
[33] www.json.org/
[34] www.mongodb.org/

4.3 Building Fill-in-the-blank and Shuffle exercises

In the previous section, we saw the mechanism used for selecting an appropriate sentence for a given pedagogical goal. *GramEx* uses such selected sentences as stem sentences to build exercise items. In this section, we describe how *GramEx* produces two types of exercises namely, Fill-in-the-blank and Shuffle exercises. Here, the exercise *question* is automatically generated from the selected sentence based on its associated linguistic properties.

FIB questions. FIB questions are built by removing a word from the target sentence and replacing it with either: a blank (FIBBlnk), a lemma (FIBLem) or a set of features used to help the learner guess the solution (FIBHint). For instance, in an exercise on pronouns, *GramEx* will use the gender, number and person features associated with the pronoun by the generation process and display them to specify which pronominal form the learner is expected to provide. The syntactic representation (cf. Figure 4.1) associated by *GraDe* with the sentence is used to search for the appropriate key word to be removed. For instance, if the pedagogical goal is *Learn Subject Pronouns* and the sentence retrieved from the generation bank is that given in (36a), *GramEx* will produce the FIBHint question in (36b) by searching for a lemma with category *cl* (clitic) and feature *func=subj* and using its gender value to provide the learner with a hint constraining the set of possible solutions.

blank out constraints

(36) a. Elle adore les petits tatous. (She loves the small armadillos)
b. _____ adore les petits tatous. (gender=fem)

Shuffle questions. Similarly to FIB questions, shuffle exercise items are produced by inspecting and using the target derivational information. More specifically, lemmas are retrieved from the list of lemma-feature pairs. Function words are (optionally) deleted. And the remaining lemmas are "shuffled" (MShuf). For instance, given the source sentence (37a), the MShuf question (37b) can be produced.

(37) a. Tammy adore la petite tatou. (Tammy loves the small armadillo)
b. tatou / adorer / petit / Tammy

Note that in this case, there are several possible solutions the learner can enter. One is with respect to morpho-syntactic information, i.e. depending on which tense and number is used by the learner. For such cases, we can either use hints as shown above to reduce the set of possible solutions to one; or compare the learner's answer to the set of output produced by *GraDe* for the semantics the sentence was

produced from (i.e. the core semantics with underspecified morphological information). Furthermore, the possible candidate solutions might vary with respect to the syntactic constructions used by the learner. If the exercise formulation states to only add determiners and morphology to give the answer, in principle, there should be no syntactic variation expected. In contrast, if the exercise formulation does not give guidelines on the expected syntax, possible correct alternative answers for the question (37b) (e.g. *C'est Tammy qu'adore la petite tatou* (It is Tammy that loves the small armadillo)) could be expected. Here, we can also rely in the fact that grammar generates different syntactic configuration for the same input semantics. Finally, the example in (37) illustrates another ambiguity issue that occurs when there is no enough information for the learner to know from the question whether it is *Tammy* who loves the *armadillo* or the other way around. The same thing occurs with the adjective, the learner does not know from the question sentence in which NP the adjective goes (although in this example the NP *La petite Tammy* would be less natural). To address these issues there are different alternative solutions. One option is to show semantic features indicating roles, however, this might force the learner to get used to this kind of semantic information. Another possibility is to collect from the sentence bank all those sentences whose transformation into Shuffle leads to the same question (e.g. collect all the sentences that after been turned into Shuffle questions lead to the same set of lemmas). For the modification ambiguity, *GramEx* could use brackets to indicate lemmas that go together into a NP, e.g. (tatou / petit) / adorer / Tammy.

4.3.1 Evaluation: correctness, variability and productivity

At this point, we carried out an experiment designed to assess the exercises produced by *GramEx*. On one hand, we want to evaluate the correctness of the exercises produced (e.g. grammaticality of the generated sentences or accuracy of the stem selection mechanism). On the other hand, we want to assess the impact of this framework in the automation of grammar exercises production. To this end, we summarise data from the experiment in different ways, as will be explained below, to provide measures of both variability and productivity. In what follows, we describe the parameters of this experiment namely, the input and the user-defined parameters constraining sentence generation; and the pedagogical goals being tested. The grammar and the lexicon resources are those mentioned in Section 4.2.3. After describing the experimental settings we discuss the results we obtain.

Pedagogical Goals

We evaluate the approach on 16 pedagogical goals[35] taken from the *Tex's French Grammar* book. For each of these goals, we define the corresponding linguistic characterization in the form of a *GramEx* query. We then evaluate the exercises produced by the system for each of these queries. The pedagogical goals tested are the following (we indicate in brackets the types of learning activity produced for each teaching goal by the system):

- Adjectives: Adjective Order (MSHUF), Adjective Agreement (FIBLEM), Prenominal adjectives (FIBLEM), Present and Past Participial used as adjectives (FIBLEM), Regular and Irregular Inflexion (FIBLEM), Predicative adjectives (MSHUF)
- Prepositions: Prepositional Infinitival Object (FIBBLNK), Modifier and Complement Prepositional Phrases (FIBBLNK)
- Noun: Gender (FIBLEM), Plural form (FIBLEM), Subject Pronoun (FIBHINT).
- Verbs: Pronominals (FIBLEM), -ir Verbs in the present tense (FIBLEM), Simple past (FIBLEM), Simple future (FIBLEM), Subjunctive Mode (FIBLEM).

GraDe's Input and User-Defined Parameters

GraDe's configuration. As mentioned in Section 4.2.1, we run *GraDe* using two main configurations. In the first configuration, *GraDe* search is constrained by an input core semantics which guides the grammar traversal and forces the output sentences to verbalise this core semantics. In this configuration, *GraDe* will only produce the temporal variations supported by the lexicon (the generated sentences may be in any simple tense i.e., present, future, simple past and imperfect) and the syntactic variations supported by the grammar for the same MRSs (e.g., active/passive voice alternation and cleft arguments).

Greater productivity (i.e., a larger output/input ratio) can be achieved by providing *GraDe* with less constrained input. Thus, in the second configuration, we run *GraDe* not on core semantics but on the full grammar. To constrain the search, we specify a root constraint which requires that the main verb of all output sentences is an intransitive verb. We also set the constraints on recursive rules so as to exclude the inclusion of modifiers. In sum, we ask *GraDe* to produce all clauses (i) licensed by the grammar and the lexicon; (ii) whose verb is intransitive; and (iii) which do not include modifiers. Since the number of sentences that can be produced under

[35] The list of defined pedagogical goals together with an extract of the automatically generated exercises is in given in Appendix A.1.

this configuration is very large, we restrict the experiment by using a lexicon containing a single intransitive verb (*chanter/To sing*), a single common noun and a single proper name. In this way, syntactically structurally equivalent but lexically distinct variants are excluded.

Input Semantics. We use two different sets of input semantics for the semantically guided configuration: one designed to test the pedagogical coverage of the system (Given a set of pedagogical goals, can *GramEx* generate exercises that appropriately target those goals?); and the other to illustrate linguistic coverage (How much syntactic variety can the system provide for a given pedagogical goal?).

Give the plural form of the noun indicated in parentheses. Pay attention to both the article and the noun.

1. Bette aime _____ . (le bijou)
2. Fiona aime _____ . (le cheval)
3. Joe-Bob aime _____ américaines. (la bière)
4. Tex n'aime pas _____ . (le choix)
5. Joe-Bob n'aime pas _____ difficiles. (le cours)
6. Tammy n'aime pas _____ . (l'hôpital)
7. Eduard aime _____. (le tableau)
8. Bette aime _____ de Tex. (l'oeil)
9. Tex aime _____ français. (le poète)
10. Corey aime _____ fraîches. (la boisson)
11. Tammy aime _____ américains. (le campus)
12. Corey n'aime pas _____ . (l'examen)

Figure 4.3: Grammar exercises from the *Tex's French Grammar*Textbook

The first set (D1) of semantic representations contains 9 items representing the meaning of example sentences taken from the *Tex's French Grammar* textbook. For instance, for the first item in Figure 4.3, we use the semantic representation *L1:named(J bette_ n) A:le_ d(C RH SH) B:bijou_ n(C) G:aimer_ v(E J C)*. With this first set of input semantics, we test whether *GramEx* correctly produces the exercises proposed in the *Tex's French Grammar* book. Each of the 9 input semantics corresponds to a distinct pedagogical goal.

The second set (D2) of semantic representations contains 22 semantics, each of them illustrating distinct syntactic configurations namely, intransitive, transitive and ditransitive verbs; raising and control; prepositional complements and modifiers; sentential and prepositional subject and object complements; pronominal verbs; predicative, attributive and participial adjectives. With this set of semantics, we introduce linguistically distinct material thereby increasing the variability of the exercises i.e., making it possible to have several distinct syntactic configurations for the same pedagogical goal.

Evaluation, Results and Discussion

Using the experimental setup described in previous sections, we evaluate *GramEx* on the following points:

- Correctness: Are the exercises produced by the generator grammatical, meaningful and appropriate for the pedagogical goal they are associated with?
- Variability: Are the exercises produced linguistically varied and extensive? That is, do the exercises for a given pedagogical goal instantiate a large number of distinct syntactic patterns?
- Productivity: How much does *GramEx* support the production, from a restricted number of semantic input, of a large number of exercises?

Correctness. To assess correctness, we randomly selected 10 (pedagogical goal, exercise) pairs for each pedagogical goal in Section 4.3.1 and asked two evaluators to say for each pair whether the exercise text and solutions were grammatical, meaningful (i.e., semantically correct) and whether the exercise was adequate for the pedagogical goal. The results are shown in Table 4.2 and show that the system although not perfect is reliable. Most sources of grammatical errors are cases where a missing word in the lexicon fails to be inflected by the generator. Cases where the exercise is not judged meaningful are generally cases where a given syntactic construction seems odd for a given semantics content. For instance, the sentence *C'est Bette qui aime les bijoux* (It is Bette who likes jewels) is fine but *C'est Bette qui aime des bijoux* although not ungrammatical sounds odd. Finally, cases judged inappropriate are generally due to an incorrect feature being assigned to a lemma. For instance, *avoir* (To have) is marked as an -ir verb in the lexicon which is incorrect.

Grammatical	Meaningful	Appropriate
91%	96%	92%

Table 4.2: Exercise Correctness tested on 10 randomly selected (pedagogical goal, exercise pairs)

We also asked a language teacher to examine 70 exercises (randomly selected in equal number across the different pedagogical goals) and give her judgment on the following three questions:

- A. Do you think that the source sentence selected for the exercise is appropriate to practice the topic of the exercise? Score from 0 to 3 according to the degree (0 inappropriate - 3 perfectly appropriate)

- B. The grammar topic at hand together with the complexity of the source sentence make the item appropriate for which language level? A1,A2,B1,B2,C1[36]
- C. Utility of the exercise item: ambiguous (not enough context information to solve it) / correct

For Question 1, the teacher graded 35 exercises as 3, 20 as 2 and 14 as 1 pointing to similar problems as was independently noted by the annotators above. For question B, she marked 29 exercises as A1/A2, 24 as A2, 14 as A2/B1 and 3 as A1 suggesting that the exercises produced are non trivial. Finally, she found that 5 out of the 70 exercises lacked context and were ambiguously phrased.

Nb. SP	1	2	3	4	5	6	7	8	9	10	14	21
(S,G)	3	33	16	7	2	4	6	1	4	1	2	6

Table 4.3: Variability: Distribution of the number of distinct sentential patterns that can be produced for a given pedagogical goal from a given input semantics.

Variability. For any given pedagogical goal, there usually are many syntactic patterns supporting learning. For instance, learning the gender of common nouns can be practiced in almost any syntactic configuration containing a common noun. We assess the variability of the exercises produced for a given pedagogical goal by computing the number of distinct morpho-syntactic configurations produced from a given input semantics for a given pedagogical goal. We count as distinct all exercise questions that are derived from the same semantics but differ either in syntax (e.g., passive/active distinction) or in morphosyntax (determiner, number, etc.). Both types of differences need to be learned and therefore producing exercises which, for a given pedagogical goal, expose the learner to different syntactic and morpho-syntactic patterns (all involving the construct to be learned) is effective in supporting learning. However, we did not take into account tense differences as the impact of tense on the number of exercises produced is shown by the experiment where we generate by traversing the grammar rather than from a semantics (as explained below). Table 4.3 shows for each (input semantics, teaching goal) pair the number of distinct patterns observed. The number ranges from 1 to 21 distinct patterns with very few pairs (3) producing a single pattern, many (33) producing two patterns and a fair number producing either 14 or 21 patterns.

[36] $A1$, $A2$, $B1$, $B2$ and $C1$ are reference levels established by the Common European Framework of Reference for Languages: Learning, Teaching, Assessment (cf. `http://en.wikipedia.org/wiki/Common_European_Framework_of_Reference_for_Languages`) for grading an individual's language proficiency.

Pedagogical Goal	FIBLem	FIBBlnk	MShuf	FIBHint
Preposition	—	28	—	—
Prepositions with infinitives	—	8	—	—
Subject pronouns–il	—	—	—	3
Noun number	11	—	—	—
Noun gender	—	49	—	—
Adjective order	—	—	30	—
Adjective morphology	30	—	—	—
Adjectives that precede the noun	24	—	—	—
Attributive Adjectives	—	—	28	—
Irregular adjectives	4	—	—	—
Participles as adjectives	4	—	—	—
Simple past	78	—	—	—
Simple future	90	—	—	—
-ir verbs in present	18	—	—	—
Subjunctive mode	12	—	—	—
Pronominal verbs	12	—	—	—
Total	236	78	30	3

Table 4.4: Number and Types of Exercises Produced from the 28 input semantics

Nb. Ex.	1	2	4	5	6	12	17	18	20	21	23	26	31	37	138
Nb. Sem	1	4	6	1	4	3	1	1	1	1	1	1	1	1	1

Table 4.5: Exercise Productivity: Number of exercises produced per input semantics.

Nb. PG	1	2	3	4	5	6
Nb. sent	213	25	8	14	10	6

Table 4.6: Pedagogical Productivity: Number of Teaching Goals the source sentence produced from a given semantics can be used for.

Productivity. When used to generate from core semantic representations (cf. Section 4.3.1), *GramEx* only partially automates the production of grammar exercises. Semantic representations must be manually input to the system for the exercises to be generated. Therefore, the issue arises of how much *GramEx* helps automating exercise creation. Table 4.4 shows the breakdown of the exercises produced per teaching goal and activity type. In total, *GramEx* produced 429 exercises out of 28 core semantics yielding an output/input ratio of 15 (429/28). Further, Table 4.5 and 4.6 show the distribution of the ratio between (i) the number of exercises produced and the number of input semantics and (ii) the number of teaching goals the source sentences produced from input semantics i can be used for. Table 4.6 (pedagogical productivity) shows that, in this first experiment, a given input semantics

can provide material for exercises targeting up to 6 different pedagogical goals while Table 4.5 (exercise productivity) shows that most of the input semantics produce between 2 and 12 exercises[37].

When generating by grammar traversal, under the constraints described in Section 4.3.1, from one input 90 exercises are generated targeting 4 different pedagogical goals (i.e. 4 distinct linguistic phenomena).

4.4 Transformation-based grammar exercises

In the previous section, we saw how *SemTAG* based generation permits producing grammar exercise items (question and solution) of type FIB and Shuffle. Here, we focus on the generation of transformation-based grammar exercises. In this type of exercises the question and the solution are two sentences that are related by a syntactic transformation. For instance, in (38) the question (Q) is an active sentence and the solution (S) its passive form counterpart.

(38) Q: *C'est Tex qui a fait la tarte.* (It is Tex who has baked the pie.)
S: *C'est par Tex que la tarte a été faite.* (It is Tex by whom the pie has been baked.)

Our approach to generate these pairs of related sentences relies on the linguistic information associated to the sentences by the generator, in particular, the derivation trees. We use the generation bank B consisting of tuples $(S_i, L_i, \sigma_i, \tau_i)$ and the constraint language introduced in Section 4.2.1 for selecting stem sentences S_i. But this time, for the analysis of transformations we make use of the fourth component τ_i, i.e. the derivation tree associated with the sentence S_i. Indeed, tree filters are used to retrieve from the generation bank those sentences that pair stem sentences. Retrieved pairs provide the question and the solution of a given transformation exercise. These tree filters are defined on pairs of derivation trees and make use of the rich linguistic information associated by our generator with those derivation trees.

As illustrated by the exercise item in (39) extracted from the *Tex's French Grammar* book, there are two major things we need to take care about when generating pairs of transformationally related sentences. That is, (i) the informational content of the sentences and (ii) the fact that the syntax of the solution sentence should be as close as possible to that of the question sentence.

[37] If the input semantics contains a noun predicate whose gender is underspecified, the exercise productivity could be doubled. This is the case for 4 of the input semantics in the dataset D2; i.e. an input semantics containing the predicates tatou_n(C) petit_a(C) will produce variations such as: *la petite tatou* (the small armadillo (f)) and *le petit tatou* (the small armadillo (m)).

(39) Tammy : *Tex aime Bette? (est-ce que)* (Tex loves Bette?)
correct answer: *Est-ce que Tex aime Bette?* (Does Tex love Bette?)
your answer: *Est-ce que Tex aime Bette?* (Does Tex love Bette?)

If we had entered as solution the sentence *Est-ce que Bette est aimée par Tex?* (Is Bette loved by Tex?) the exercise would have being considered as incorrect. The learner is expected to enter the transformed sentence following closely the syntactic configuration of the question sentence. Regarding the informational content of the solution sentence, if the learner would have entered the answer *Est-ce que Bette aime Tex?* (Does Bette loves Tex?) it would have also been taken as incorrect. While the answer uses the question form *est-ce que* the meaning is not the same as that of the question sentence.

In what follows, we will see how the approach we propose based on *SemTAG* derivation trees addresses the generation of the type transformation-based grammar exercises discussed so far. We start by linking this approach to related work on generating or deriving syntactic reformulations of a given sentence. Then, we review the linguistic information present in the derivation trees and give some intuitions of how this information is used by the tree filters. We go on to motivate the use of derivation trees as a structure on which to base the identification of transformationally related sentences. Finally, we present the derivation tree filters used to identify pairs of transformationally related sentences.

4.4.1 Related work on sentence reformulation

In linguistics, transformations ([Harris, 1957; Chomsky, 1957]) model recurrent linguistic relations between sentence pairs. For instance, a transformation can be used to define the relation between the active and the passive voice version of the same sentence. Formally, transformations were stated as tree-transducers on phrase structure trees and they defined either structure changing or structure building (generalised transformation) operations.

In computational linguistics, transformations and more generally, structure changing and structure building rules have been used in such tasks as text simplification ([Siddharthan, 2010]), text summarising ([Cohn and Lapata, 2009]) and question generation ([Piwek and Boyer, 2012]). In these approaches however, the transformation relation is not necessarily defined on phrase structure trees. For instance, for the question generation task, Yao *et al.* (2012) have argued that Assertion/WH-Question transformations are best defined on semantic representations. Conversely, for text simplification, Siddharthan (2010) has convincingly shown that dependency

trees are better suited as a representation on which to define text simplification rules than both phrase structure trees and semantic representations.

Siddharthan (2011) presents a user evaluation comparing different re-generation approaches for sentence simplification. He notes in particular that annotators preferred those transformations that are closer in syntax to the original sentence. To achieve this, rules for word ordering are either added to the transform rules or coded as constraints within the input to a generator. In contrast, in our approach, syntactic similarity can be deduced by tree comparison using the rich linguistic information associated by the generator with the FB-LTAG derivation trees.

Chandrasekar and Srinivas (1997) describe an algorithm by which generalised rules for simplification are automatically induced from annotated training material. Similar to our work, their approach makes use of TAG derivation trees as a base representation. Using a corpus of complex sentences parsed and aligned with the corresponding simplified sentences, the tree comparison algorithm they propose permit inducing simplification rules between dependency trees derived from TAG derivation trees. Although similar to our approach, Chandrasekar and Srinivas's (1997) proposal differs from ours in several ways. First, while we focus on transformations, they work on simplifications relating e.g., a sentence containing a relative clause to two base clauses. Second, the trees on which they define their transformations are reconstructed in a rather ad hoc manner from the TAG derivation trees and from information extracted from the TAG derived trees. In contrast, we make use of the derivation trees produced by the *GraDe* algorithm. Third, while their work is limited to sentences containing relative clauses, we consider a wider range of transformations. Fourth, their approach targets the automatic acquisition of simplification rules while we manually define those.

4.4.2 SemTAG derivation trees

In Section 2.2, we introduced the *SemTAG* grammar and, in particular, its derivation trees (cf. Section 2.2.3). We also described the linguistic information present in the *SemTAG* derivation trees produced by our generator (cf. Section 4.2.1). Here, we give some intuitions about how this linguistic information will be used in the generation of pairs of syntactically related sentences.

Figure 4.4 shows a small FB-LTAG grammar and the derivation tree associated with the sentence *C'est Tammy qui fait la tarte* (It is Tammy who bakes a pie). This example derivation tree illustrates the additional information[38] contained in deriva-

[38] In Figure 4.4, edge labels are omitted for simplicity.

tion trees produced by *GraDe*. Nodes are labelled not only with the name of an elementary tree but also with the lemma anchoring that tree, the feature structure associated with the anchor of that tree and the *tree properties* of that tree.

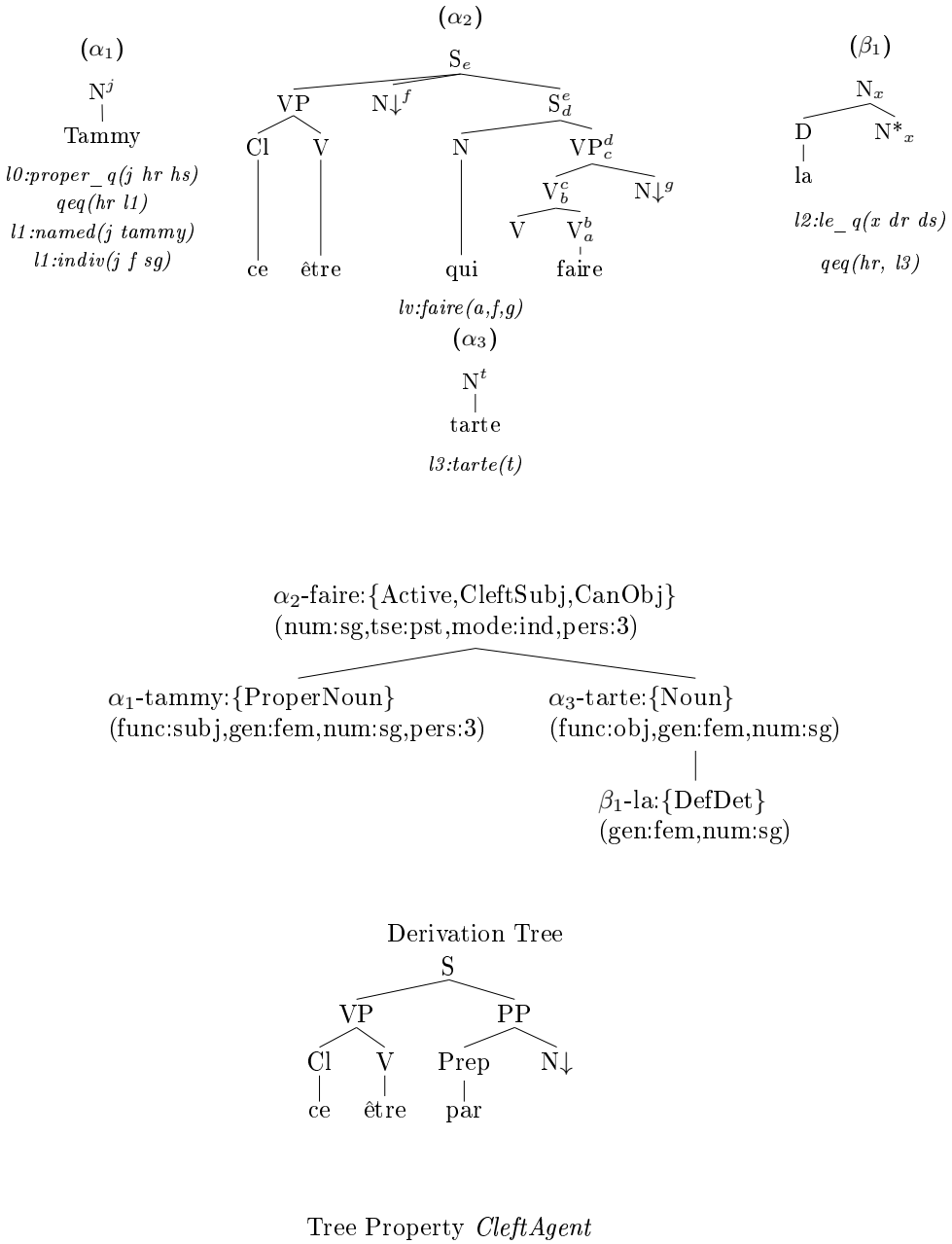


Figure 4.4: Grammar, Derivation Tree and Example Tree Property (Bottom right) for the sentence *C'est Tammy qui fait la tarte* (It is Tammy who bakes the pie)

We use feature structure information in three main ways. To identify the gram-

matical function of an argument, to verify that two transformationally related sentences are syntactically and morpho-syntactically identical up to the transformed part and to verify morpho-syntactic constraints on the transformed parts.

In Figure 4.4, the *func* feature indicates the function of the two arguments of the verb: *Tammy* is the subject and *tarte* (pie) the object of *faire* (to bake). The tense-aspect morphological features (*tse* and *aspect*) for the verb *faire* permit verifying temporal relations between a sentence and its transform. Similarly, the verb feature *mode* for verifying the mode of two sentences. The gender, number and person (*gen, num, pers*) features in, for example, a noun argument of a verb (*tarte* in Figure 4.4) allow verifying that the noun argument is the same or replaced by an adequate syntactic element, e.g. a pronoun, *Tammy* $\rightarrow$ *elle* (she –nominative case).

Recall from Section 2.2.5, tree properties are abstractions over tree descriptions. They name the tree descriptions that were used to build the FB-LTAG elementary trees. Thus, for instance, the tree property *CleftAgent* names the tree description appearing at the bottom right of Figure 4.4; and the elementary tree α_2 in Figure 4.4 is associated with the tree properties *Active, CleftSubj,CanObj* indicating that this tree was built by combining the tree descriptions named *Active, CleftSubj* and *CanObj*.

4.4.3 Why Derivation Trees?

While providing detailed information about the syntax and the informational content of a sentence, FB-LTAG derivation trees provide both a more abstract description of this information than derived trees and a richer representation than semantic formulae.

Figure 4.5 illustrates the difference between derived and derivation trees by showing those trees for the sentences *C'est Tex qui a fait la tarte* (It is Tex who baked the pie) and *C'est par Tex que la tarte a été faite* (It is by Tex that the pie was baked). While the derived trees of these two sentences differ in their overall structure (different structure, different number of nodes), their derivation trees are identical up to the tree properties of the verb. Moreover, the tree properties of the active ({*Active,CleftSubj,CanObj*}) and of the passive ({*passive,cleftAgent,canSubj*}) verb capture the changes in argument and verb realisation typical of a passive transformation. In other words, derivation trees provide a level of description that is simpler (less nodes) and that better supports the identification of tranformationally related sentences (more similar configurations and explicit description of changes in argument and verb realisation).

Derivation trees are also better suited than semantic formulae to capture transfor-

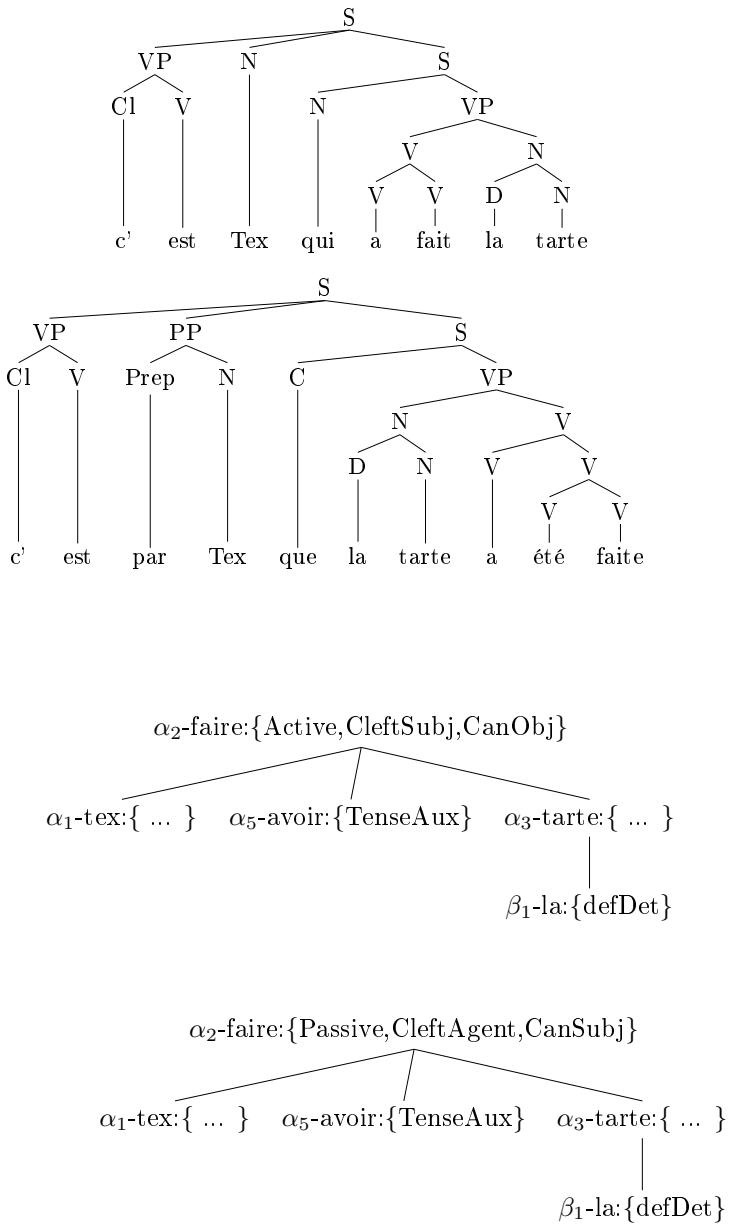


Figure 4.5: Derived (top) and Derivation (bottom) Trees for the active voiced sentence *C'est Tex qui a fait la tarte* (It is Tex who baked the pie) and its passive variant

mations as, in some cases[39], the semantic representations of two transformationally

[39] Whether two syntactically distinct sentences share the same semantics depends on the grammar. In the grammar we use, the semantic representations aim to capture the truth conditions of a sentence not their pragmatic or informational content. As a result, Passive/Active variations do share the same semantics.

related sentences may be identical. For instance, in our grammar, Active/Passive, canonical/inverted subject and cleft/non cleft argument variations are assigned the same semantics. As illustrated in Figure 4.5, for those cases, the tree properties labelling the derivation trees provide a direct handle for identifying sentences related by these transformations.

4.4.4 Tree filters for transformation related sentences

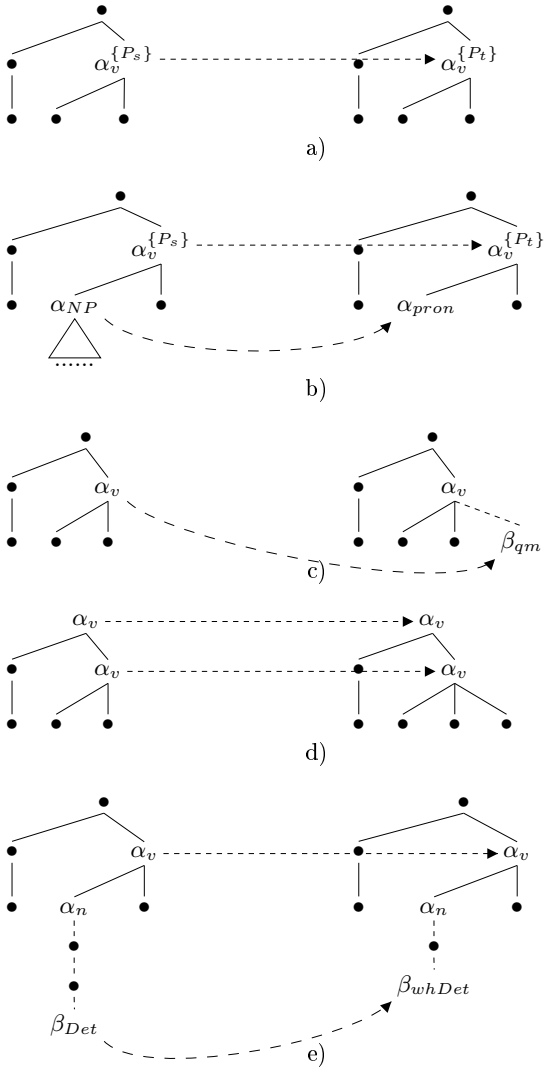


Figure 4.6: Tree filter types (tree schemas on the left depict source sentence derivation trees and those to their right their transform).

To identify transformationally related sentences, we define tree filters on deriva-

tion trees. These filters make use of all the information present in the FB-LTAG derivation trees produced by *GraDe* namely, the tree names, the lemmas, the feature structures and the tree properties labelling the nodes of these trees.

Figure 4.6 shows the general filtering patterns we used to handle four types of transformations used in language learning: Active/Passive, NP/pronoun (pronominalisation), NP/Wh-NP (WH-questions) and Assertion/Yes-No questions.

Filters (a) and (d) are used for the Active/Passive and for the canonical/inverted subject variations. Filter (a) relates two trees which are identical up to one node differing in its tree properties. It applies for instance to the derivation trees shown in Figure 4.5. Filter (d) is used for cases such as *John wants Mary to like him / John wants to be liked by Mary* where the two derivation trees differ both in the tree properties assigned to *want* (*CanSubj, CanObj, SentObj* $\leftrightarrow$ *CanObj, SentObj*) and in the tree properties assigned to *like* (*InfSubj, CanObj* $\leftrightarrow$ *InfSubj, CanAgent*); and where an additional node is present due to the presence of the pronoun *him* in the active sentence and its absence in the passive variant.

Filter (b) is used for the NP/Pronoun transformation and relates two trees which in addition to having one node with different tree properties also differ in that an NP node and its subtree maps to a pronoun node.

Filter (c) relates two trees which are identical up to the addition of an auxiliary tree of type β_{qm}. As we shall see below, this is used to account for the relation between an assertion and a question including a question phrase (i.e., *n'est ce pas / Isn't it*, *est ce que*, inverted *t'il* or question mark).

Finally, Filter (e) is used for the assertion/wh-question transformation and matches pairs of trees such that an NP containing n modifiers in one tree becomes a WH-NP with any number of these n modifiers in the other tree.

We now discuss in more detail the derivation tree filters specified for each type of transformations.

4.4.5 Meaning Preserving Transformations

In *SemTAG* semantic representations aim to capture the truth conditions of a sentence not their pragmatic or informational content. As a result, some sentences with different syntax share the same semantics. For instance, all sentences in (40b) share the semantics in (40a).

(40) a. L0:proper_q(C HR HS) L1:named(C tammy) L1:indiv(C f sg) qeq(HR L1) L3:love(EL TX C) L3:event(EL pst indet ind) L4:proper_q(TX HRX HSX) L5:named(TX tex) L5:indiv(TX m sg) qeq(HRX L5)

b. *Tex loves Tammy, It is Tex who loves Tammy, It is Tammy whom Tex loves, Tammy is loved by Tex, It is Tammy who is loved by Tex, It is by Tex that Tammy is loved, etc.*

The syntactic and pragmatic differences between these semantically identical sentences is captured by their derivation trees and in particular, by the tree properties labelling the nodes of these derivation trees. More generally, Active/Passive sentence pairs, canonical/cleft (e.g., *Tex loves Tammy / It is Tex who loves Tammy* and Canonical/Inverted Subject variations (e.g. *C'est Tex que Tammy adore / C'est Tex qu'adore Tammy*) may lead to derivation trees of identical structure but distinct tree properties. In such cases, the transformationally related sentence pairs can therefore be captured using the first type of derivation filter i.e., filters which relate derivation trees with identical structure but distinct tree properties. Here, we focus on the Active/Passive variation.

The differences between an active voice sentence and its passive counterpart include lexical, morphological and syntactic differences. Thus for instance, (41a) differs from (41b) in that the verb agrees with the proper name *Tammy* rather than the pronoun *il*; the clitic is in the oblique case (*lui*) rather than the nominative (*il*); the subject NP *Il* has become a PP headed by the preposition *par*; the passive auxiliary *être* and the preposition *par* have been added to support the passive voice construction.

(41) a. *Il regarde Tammy* (He watches Tammy)
b. *Tammy est regardée par lui* (Tammy is watched by him)

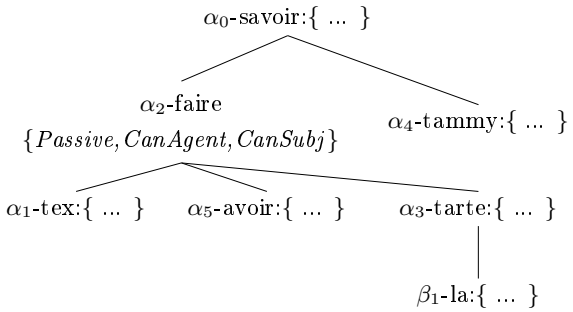
In [Siddharthan, 2010], these variations are handled by complex node deletion, lexical substitution, insertion, and node ordering rules. By contrast, to identify `Active / Passive` variations, we search for pairs of derivation trees that are related by an Active/Passive derivation tree filter namely, a filter that relates two trees which are identical up to a set of tree properties labelling a single node pair. We specify as many Active/Passive tree property patterns as there are possible variations in argument realisations. For instance, for a transitive verb, some of the defined tree property patterns are as follows:

Active/Passive Tree Property Patterns

{*Active, CanSubj, CanObj*} ↔ {*Passive, CanSubj, CanAgent*}
{*Active, CliticSubj, CanObj*} ↔ {*Passive, CanSubj, CanAgent*}
{*Active, WhSubj, CanObj*} ↔ {*Passive, InvertedSubj, WhAgent*}
{*Active, RelSubj, CanObj*} ↔ {*Passive, CanSubj, RelAgent*}
{*Active, CleftSubj, CanObj*} ↔ {*Passive, CanSubj, CleftAgent*}

In sum, in our approach, the possible differences in morphological agreement between active and passive sentences are accounted for by the grammar; differences in argument realisation (Object/Subject, Subject/Agent) are handled by the tree filters; and lexical differences due to additional function words fall out of the FB-LTAG coanchoring mechanism.

As should be clear from the derivation tree below, our approach supports transformations at any level of embedding. For instance, it permits identifying the pair *Tammy sait que Tex a fait la tarte / Tammy sait que la tarte a été faite par Tex* (Tammy knows that Tex has baked the pie / Tammy knows that the pie has been baked by Tex).



It also supports a fine grained control of the Active/passive variants allowing both for cases with multiple variants (42a) and for transitive configurations with no passive counterpart (42b,d).

(42) a. *C'est la tatou qu'il adore*
C'est par lui que la tatou est adorée
C'est lui par qui la tatou est adorée
(It is the armadillo that he loves / It is the armadillo that is loved by him)

b. *Tex veut faire une tarte*
*** Une tarte veut être faite par Tex*
(Tex wants to bake a pie / ** A pie wants to be baked by Tex)

c. *Tex semble faire une tarte*
Une tarte semble être faite par Tex
(Tex seems to bake a pie / A pie seems to be baked by Tex)

d. *Tex mesure 1.80m*
*** 1.80m est mesuré par Tex*
(Tex measures 1.80m ** 1.80m is measured by Tex)

(42a) is accounted for by specifying a tree filter including the tree property mapping *CleftSubject* ↔ *CleftAgent* where *CleftAgent* subsumes the two types of clefts illustrated in (42a).

The lack of passive in (42b) and (42d) is accounted for by the grammar: since (42b) does not licence a passive, the starred sentence will not be generated. Similarly, because the verb *mesurer / to be X tall* is not passivable, the starred sentence in (42d) will not be produced.

4.4.6 Meaning Altering Transformations

When the content of two sentences differs, in particular, when a content word is deleted or added, the derivation trees of these sentences may have a different structure. In those cases, we use filters that relate derivation trees with distinct tree structures namely, filters (b), (c), (d) and (e) in Figure 4.6.

NP/Pronoun To handle the NP/Pronoun, we use the filter sketched in Figure (4.6b) which relates derivation trees that are identical up to an NP subtree replaced by a node labelled with a pronoun. In this way, the difference between the derivation tree of *le tatou* (two nodes) and *il* (one node) does not prevent the identification of sentence pairs such as (43a).

(43) a. *Le tatou chante*
Il chante (*Personal pronoun*)
The tatoo sings/He sings

b. *Quel tatou chante ?*
Qui chante ? (*WH-Personal Pronoun*)
Which armadillo sings?/Who sings?

NP/Wh-NP For wh-questions, the main difficulty is to account for variations such as (44) below where a complex NP with several modifiers can map to a Wh-NP with different numbers of modifiers. To capture these various cases, we use two tree filters. The first filter is similar to filter (b) in Figure 4.6 and matches NP/WH-Pronoun sentences (e.g., 44a-b where the NP *Le grand tatou avec un chapeau qui dort sous le palmier* maps to a WH-Pronoun *qui*). The second tree filter is sketched

in Figure (4.6e). It matches NP/Wh-NP sentences (e.g., 44a-c/f) where an NP matches to a WH-NP headed by a WH-Determiner, the head noun and any number of modifiers.

(44) a. Le grand tatou avec un chapeau qui dort sous le palmier ronfle.
Qui ronfle ? Quel tatou ronfle ? Quel grand tatou ronfle? Quel tatou avec un chapeau ronfle ? Quel tatou qui dort sous un palmier ronfle ? etc.
The big armadillo with a hat who sleeps under the palmtree snores/ Who snores? Which armadillo snores? Which armadillo with a hat snores? Which armadillo who sleeps snores? etc.

Yes-No Question. In French, yes/no questions can be formed in several ways:

(45) a. Le tatou chante
Le tatou chante-t-il? (*Inverted t-il*)
Est ce que le tatou chante ? (*est ce que*)
Le tatou chante? (*Intonation*)
Le tatou chante n'est ce pas? (*n'est ce pas (isn't it)*)
The armadillo sings / Does the armadillo sing? The armadillo sings? The armadillo sings doesn't it?

b. Vous chantez
Chantez-vous? (*Inverted Subject*)
You sing/Do you sing?

For cases such as (45b), we require the derivation trees to be identical up to the tree property mapping *CliticSubject* ↔ *InvertedCliticSubject*. For cases such as (45a) on the other hand, we use the filter sketched in Figure (4.6c) that is, a filter which requires that the derivation trees be identical up to a single additional node licenced by a question phase (i.e., *t'il, est ce que, n'est ce pas* or a question mark).

4.4.7 Evaluation: coverage, genericity and precision

Our first evaluation experiment in Section 4.3.1 was mainly evaluating general feasibility of the *GraDe* framework. Now, we focus on the evaluation of the approach to generate transformation-based grammar exercises. To this end, we carry out an experiment to assess the genericity, the correctness and the coverage (i.e. recall) of this approach. The grammar and lexicon used in this experiment are the same as the ones described in the evaluation in previous Section (4.3.1). In what follows, we describe the sentence set used for testing; and the results obtained.

	S	T	TF	Precision
Active/Passive	43	38	8	84
Pronominalisation	36	33	7	73
Wh-Questions	25	20	7	88
YN-Questions	24	20	5	90.5

Table 4.7: Source Sentences (S), Transformations of Source Sentences (T), Number of Filters (F) and Precision (Ratio of correct transformations)

Generated Sentences. To populate the generation bank, we input *GraDe* with 52 semantic formulae corresponding to various syntactic and semantic configurations and their interactions[40]: including all types of realisations for verb arguments (cleft, pronominalisation, relative, question arguments); Intransitive, Transitive and ditransitive verbs; Control, raising and embedding verbs; Nouns, common nouns, personal strong and weak pronouns; standard and Wh- determiners.

From these 52 semantic formulae, *GraDe* produced 5748 sentences which we stored together with their full semantics and their derivation tree.

Results. Table 4.7 summarises the results of our experiment. It indicates the number of source sentences manually selected so as to test different syntactic configurations for each type of transformation considered (S), the number of transformations found for these source sentences (T), the number of tree filters used for each type of transformation (TF) and the precision obtained (ratio of correct transformations).

The low number of tree filters relative to the number of syntactic configurations explored indicates a good level of genericity: with few filters, a transformation can be captured in many distinct syntactic contexts. For instance, for the Active/Passive transformation, 8 filters suffice to capture 43 distinct syntactic configurations.

As expected in an approach where the filters are defined manually, precision is high indicating that the filters are accurate. The generated pairs marked as incorrect by the annotator are all cases where the transformed sentence was ungrammatical; in other words, the filters were accurate.

Finally, the relatively low number of transformations found relative to the number of source sentences (e.g., 38 transforms for 43 source sentences in the Active/Passive case) is mainly due to transformed sentences that are missing from the generation bank either because the corresponding input semantics is missing or because of gaps in the grammar or the lexicon. However, for few cases missing filters were identified as well.

[40]We restrict the tense of the verb of the main clause to present and indicative mode

4.5 Comparison with previous work on (semi-)automatic grammar exercises generation

We have already initiated the discussion regarding related work on the (semi-)automatic generation of grammar exercises in Section 2.3.1. After having presented our approach we resume the discussion with a more detailed comparative view with respect to the exercise types and evaluation results.

ArikIturri ([Aldabe *et al.*, 2006]) automatically constructs grammar exercise items from a corpus annotated with syntactic and morpho-syntactic information. They retrieve sentences from this annotated corpus based on some target linguistic phenomena that is to be practiced. The types of exercises handled by the approach are Fill-in-the-blank, word formation, multiple-choice and error detection. The Fill-in-the-blank and word formation exercises correspond to our different types of Fill-in-the-blanks exercises (i.e. FIBBLNK and FIBLEM). As in our approach, the blank is identified based on morpho-syntactic information. Aldabe *et al.*'s (2006) approach support the generation of two additional exercise types namely, multiple-choice questions and error detection queries. To create distractors for multiple-choice questions, they use alternative inflections or conjugations of the words in the key phrases. These types of exercises could also be added to our *GramEx* framework. They carried out an evaluation experiment in which a language teacher manually evaluated 1350 automatically generated questions of multiple-choice and error-detection types. The percentage of the automatically generated questions accepted by the teacher was around 80%.

Similarly, in the FAST ([Chen *et al.*, 2006]) system, grammar-based multiple-choice and error detection exercise types are generated automatically. A bank of sentences is constructed from sentences gathered from the Web and each sentence is POS tagged and chunked. Exercises are generated from manually defined "test patterns". For instance, a pattern for a multiple-choice exercise type describes the surface pattern the source sentence should have and the patterns to generate the distractors. The carried out an evaluation in which 70 exercise patterns where defined based on grammar rules taken from the book[41]. From Wikipedia and the VOA (Voice Of America) corpus they extracted 3872 sentences which were transformed into 25906 multiple-choice questions and 2780 sentences were converted into 24221 error detection questions. A sub-set of these exercises was given to language teachers and students for manual evaluation. From 1359 multiple-choice questions 77% were considered correct (either without or with minor post-editing) while from 1908 error

[41] "How to prepare for the TOEFL" by Sharpe (2004)

detection questions, 80% were considered correct.

Although the architectures, resources, exercise types and size of the samples manually evaluated are different, the rate of correct generated exercises in our approach is similar to that obtained in these approaches with around 90% of the exercises considered correct by the evaluators.

Two important points distinguish our exercise generation framework from Arik-Iturri and FAST. First, the syntax and vocabulary of the exercise stems can be highly constrained using rich and fine-grained linguistic information. Second, we handle another type of exercises namely, the transformation-based exercises.

4.6 Conclusions and perspectives

In this chapter, we presented the *GramEx* framework for the (semi-)automatic generation of grammar exercises which are similar to those often used in textbooks for second language learning. Their syntax and vocabulary are controlled having as objective to make it easier for the learner to concentrate on the grammatical point or vocabulary to be learned.

Grammar exercises target a specific pedagogical goal, therefore, they are built from stem sentences that permit exercising that pedagogical goal. Stem sentences are further processed to build both the exercise question and the target solution. Concretely, the type of exercise that *GramEx* builds are Fill-in-the-blank, Shuffle and transformation-based grammar exercises.

As mentioned at the beginning of this chapter and shown throughout it, a key feature of the approach is the rich linguistic information associated by the generator with the source sentences. Moreover, the underspecification mechanism of our generator together with the paraphrastic grammar make possible to obtain morpho-syntactic and syntactic variations from a single input. In other words, we do not need to fully specify a different input for each different (morpho-)syntactic output configuration. This is an important aspect regarding the (semi-)automatic production of material for grammar exercise generation.

We carried out two evaluation experiments which allowed us to assess the impact of the generation based approach on the automation of grammar exercises generation (variability and productivity) as well as the correctness of the proposed mechanisms for stem generation and exercise item building. There is some manual work required to provide core semantic inputs and afterwards revising the generated material. Nevertheless, there is a wide number of exercises that are automatically generated and whose completely manual authoring would have being

tedious and a more time consuming task. Issues related to over-generation (e.g. "ungrammatical" cases pointed out by the annotators for some of the stem sentences) and under-generation (e.g. missing sentences detected when finding transformationally related sentence pairs) can be addressed by making use of existing tools for grammar debugging and error mining ([Gardent and Kruszewski, 2012; Gardent and Narayan, 2012]).

In particular, as shown in the evaluation, the generation of transformation-based exercises suffers from a low coverage. The reasons for this are twofold facts: missing counterpart sentences in the generation bank and missing tree filters. The latter can be improved by applying machine learning techniques in a similar way as in [Chandrasekar and Srinivas, 1997] for inferring tree filters (or transformation rules) from our corpus of FB-LTAG derivation trees annotated with transformation relations.

GramEx input

In this chapter, we have considered as input to the generation process a set of *GraDe* constraints (cf. Section 4.2.1). Concretely, in the evaluation experiments, we have used both types of *GraDe* user-defined constrainted, either syntactic constraints or a core semantic specification. Recall that we mean by "core semantics" core predicate/argument and modifier/modifiee relationships. When generating from a core semantics, *GraDe* will first automatically complete, or "enrich", this semantics (possibly in different ways) as dictated by the underlying grammar and lexicon. Because this enrichment process is based on the *SemTAG* grammar traversal, the obtained meaning representations are well-formed with respect to the type of semantic formulae expected by the realiser. In the future, we foresee two additional ways in which the input formulae to *GramEx* could be produced. One possibility is through an interface which lets the language teacher enter basic core content by using some templates. These templates could then be input to *GraDe* to automatically generate a semantic formula compatible with the realiser. Another possibility is to derive meaningfull core semantics from some existing knowledge source. For instance, a knowledge base as in the case in I-FLEG ([Amoia *et al.*, 2012; Denis *et al.*, 2012]). In this language game, there is a Description Logics (DL) describing the objects of the 3D world. I-FLEG's architecture includes a content selection module that extracts facts from the underlying knowledge base (the working of the content selection module is explained in [Denis *et al.*, 2012]). Given an object selected by the learner, the content selection module automatically extracts a set of facts describing it and which permit creating an exercise corresponding to a given pedagogical goal. These facts could be used as input to *GraDe*, and in turn to *GramEx*.

The *GramEx* framework presented in this chapter could be extended in different ways. A follow up in the transformation-based grammar exercises is the generation of more complex pairs of transformationally related sentences such as those in (46). The first one (46a) presents a lexico-syntactic transformation while the second (46b) involves a change in the phrase category (from sentence to noun phrase).

(46) a. *Tom ate because of his hunger / His hunger caused Tom to eat*
b. *Tammy loves the armadillo / The armadillo that Tammy loves*

The semantic content in the sentence bank could be further exploited by grouping meaning representations to produce aggregated phrases as proposed in [Williams and Power, 2010] or by discovering rhetorical relations as explained in [Power, 2011]. The example in (47a-b) shows a pair of sentences that could be generated by grouping and aggregating meaning representations from the sentence bank. In (48) both clauses stand in an elaboration or exemplification relation. Although we could directly write an input semantics for the sentences in (47b) and (48), by applying these techniques, the teacher only needs to author simple clauses from which complex sentences could be automatically derived.

(47) a. *Tammy mange une pomme.* (Tammy eats an apple)
Tex mange une pomme. (Tex eats an apple)
La petite tatou mange une pomme. (The small armadillo eats an apple)
b. *Tammy, Tex et la petite tatou mangent une pomme.*
(Tammy, Tex and the small armadillo eat an apple)

(48) a. *Tex travaille a l'université; Tex est professeur.*
(Text works at the university; he is a professor)

Another interesting extension to the *GramEx* approach would be to provide each exercise with some context as is frequently done in textbooks. It would be interesting to explore, for instance, how to precede the exercise question with an introductory sentence providing a more natural setting and minimizing the risk of ambiguity. For instance, in a pedagogical goal for *Learning pronouns* (e.g. 49) a Fill-in-the-blank exercise could be produced. However, the exercise item should provide hints to the learner as shown in (49c) because otherwise an exercise without these hints might result ambiguous as is the case of (49b) [42]. By providing a text with more context as in the case of the sentence (49d), the gender of the pronoun could be inferred from the referred noun phrase (i.e. *Tex and Tammy*).

[42] Note that here it is not necessary to give hints for the number because it can be inferred from the conjugation of the verb.

(49) Fill in the blank with the correct subject pronoun: je, tu, il, elle, on, nous, vous, ils, elles.

a. *Ils se sont rencontré à l'université.* (They have met at the university)

b. *Q:* _____ *se sont rencontré à l'université.*

c. *Q:* _____ *se sont rencontré à l'université. (gen=m)*

d. *Tex et Tammy sont des tatous. Ils se sont rencontré à l'université.*
(Tex and Tammy are armadillos. They have met at the university)

e. *Q: Tex et Tammy sont des tatous.* _____ *se sont rencontré à l'université.*

Chapter 5

Conclusions

5.1 Summing up and concluding

In this thesis, the common thread was the question of "how natural language generation techniques can contribute to computer assisted language learning". In what follows, we summarise and draw our conclusions on each particular line of work we pursued, *optimising surface realisation* and *generation for language learning*, to address this question.

5.1.1 On surface realisation optimisation

By constructing derivation trees rather than derived trees we provide a new surface realisation algorithm for FB-TAG, RTGen, that provides packing and structure sharing. To do this, we made use of a translation from FB-TAG to FB-RTG describing the derivation trees of the original FB-TAG grammar. This encoding permitted varying the working of the algorithm by providing a left corner version of the FB-RTG grammar and the possibility of transferring from the FB-TAG to the FB-RTG grammar different sets of features. We evaluated RTGen with different combinations thereof and found that the left-corner with selected blocking features, i.e. RTGen-selective, was the most efficient one. These results are not surprising. On one hand, in the left corner mode, top/bottom unifications occur earlier and instantiated features are available earlier too. On the other hand, by using only those features that are in the grammar to block certain structures combinations, RTGen-selective avoids overgeneration while unifications are cheaper to compute. However, when the indexing according to semantic arguments and the blocking of the proliferation of intermediate incomplete structures mechanisms are integrated into RTGen, the difference in performance between using the left-corner version or not disappears. Even though, the

latter optimisations need to be tested more extensively, the results obtained on the available data already provide some positive evidence of better performance. This also coincides with previous work. First, in their detailed comparative evaluation, Carroll and Oepen (2005) already report the benefits of reducing the search space by blocking intermediate incomplete structures. Second, indexing avoids the prediction and the attempts of combination of items which would otherwise take place if relying solely on possibly uninstantiated features in the feature structures.

RTGen's forest generation algorithm could be combined with different filtering techniques (e.g. polarity filtering or supertagging) before the tree combination phase. It could also be combined with a language model to prune possible solutions during tree combining or at the end of the unpacking phase (possibly extending what was done for intersective modifiers ordering, Section 3.2.2).

5.1.2 On generation for language learning

We showed one way of exploiting NLG for language learning: the generation of vocabulary and syntax controlled grammar exercises. The *SemTAG* based generation approach permits defining a mechanism to select stem sentences that satisfy certain linguistic properties. This is an important ingredient in supporting teaching where input is provided according to the learner's language development. Concretely, the selection mechanism we proposed permits selecting stems sentences that can be constrained with respect to the pedagogical goal (e.g. grammar point to learn/practice) and the general syntactic configuration preferred or disallowed (e.g. syntactic configuration according to learners level).

The boolean constraint language based on *SemTAG*'s linguistic properties for sentence selection, includes constraints over morpho-syntactic features in feature structures, combination of features and tree properties, disjunctions and negations thereof. One useful feature of our selection mechanism is that by means of "negated linguistic properties" it is possible to exclude certain linguistic constructions instead of listing all alternative desired ones. Our approach implements a "generate and select" strategy rather than "fully specify and generate". This was the most adequate way given that in our application to exercises generation, it might be often the case that is not a single syntactic configuration that can be used but possible set thereof. Currently, queries for sentence selection are manually specified and require a depth knowledge of the grammar. It could be interesting to investigate how to automatically guide the edition of such queries and to highlight queries that are unsatisfiable.

Grammar features, or more precisely, linguistic properties in *SemTAG*, could be

mapped to language levels. Then, *GramEx* could select stem sentences based on the linguistic phenomena required by a given pedagogical goal (specified manually for a given exercise activity) and the "degree of difficulty" of the sentence. This degree of difficulty could be automatically computed over the associated linguistic properties and their assigned language level. In addition, the degree of difficulty associated with a whole exercise item could be computed in a similar way and thus be exploited to gradually deliver exercises.

In a more detailed manner, the linguistic information encoded in the generation grammar, and in turn, associated to the generated sentences and exercise items could be further exploited for learner modelling à la Michaud and McCoy (2000). That is, an ICALL application integrating *GramEx*, such as I-FLEG, can keep track of the grammar points (encoded as *SemTAG* linguistic properties) mastered by each learner. Then, *GramEx* could exploit this information for the selection of stem sentences. Further, the application could deliver exercices to the learners based on the acquired grammar points, what they need to reinforce and probably some activity sequencing model.

As for the learning activity types, we showed that the underspecification mechanisms of the generator together with the rich linguistic information provided by the *SemTAG* grammar permits the (semi-)generation of a wide number of grammar exercise types with varied syntax (Variability) from few input (Productivity). We have examined *GramEx* exercises and we found that the exercises produced are mostly correct both linguistically and pedagogically (Correctness). Furthermore, we have shown that our approach can generate both exercises that are with a unique answer and exercises with a small set of correct alternative answers.

5.2 Future work and research directions

At this point, as a final step in taking stock of the work of this thesis we discuss what would be coming next. We point out to future work that needs to be done to further complement or extend the approaches developed in this thesis and suggest directions for further research.

5.2.1 Surface realisation learning by generating

Some techniques to prune the search space are informed by the probability of lexical dependencies from observed data, in data-driven approaches, or by syntactic information about the lexical items encoded in the grammar in symbolic approaches (e.g. valency and categorial information in polarity filtering).

In the absence of annotated data, one possibility would be to learn (store for future use) this information from previous runs. This idea may be seen as an instance of a technique called Explanation-Based Learning (EBL, [DeJong and Mooney, 1986]). This technique, popular during the 80's and 90's, aims at inferring generalised rules from observed examples in order to speed up further processing or extend a given theory. In particular, within the context of constraint satisfaction problems (CSPs) the learned elements are called *nogoods* (or conflicts). They are fruitless trials in the search that are recorded so as not to attempt to use them again when backtracking or in future computations ([Dechter, 1990]). Indeed, this technique has already been used in the context of TAG parsing ([Srinivas and Joshi, 1995]), where models of successful parses are learned. In the RTGen algorithm, it would be interesting to explore how information about successful/unsuccessful item combinations could be stored for guiding the search in future realisations.

5.2.2 Widening RTGen domain of application

A straightaway step is to complete the small evaluation performed with the RTGen extensions (semantic argument indexing and blocking of intermediate structures presented in Section 3.2.2) with a comparatively larger scale evaluation as carried out for the base algorithm. It would also be interesting to evaluate RTGen performance on the generation of "real world" sentences, for instance, generating from the Surface Realisation Shared Task ([Belz *et al.*, 2011]) deep inputs. Though, as we discussed in Section 3.3.2, we would need to first develop an input converter from the SR task input to the input format expected by our realiser. A benefit of this evaluation is that it would permit debugging the grammar in terms of over-generation (and possibly under-generation, though, in this cases, it would not be clear how to distinguish from lack in grammar coverage to "non expected" input). Further, we could probably generate paraphrases (and grammar exercises) from "real world" sentences but with associated rich linguistic information for language learning.

Another direction to explore is how *SemTAG* and RTGen surface realisation techniques can be exploited for generation in the context of *Conceptual Authoring* ([Hallett *et al.*, 2007]). In conceptual authoring, text and meaning representations are linked, and editions on the text trigger modifications in the corresponding underlying meaning and viceversa. The text is dynamically generated (from the underlying meaning representation) and modified (by the user using operations –e.g. insertion, deletion and replacement of spans of text in [Franconi *et al.*, 2010]– supported in the specific interface).

SemTAG seems to be a natural candidate to support conceptual authoring, as

it systematically relates text, syntax and semantics. But the question about how to efficiently support dynamic (or incremental) content realisation due to deletion, replacement and addition of text arises. In some cases, adding text would involve just a simple coordination while in other cases the left (and right) context will have to bear structural and/or lexico-syntactic modifications ([Smedt, 1990]). So, how to determine the parts of the existing text that are affected? Which are the lexico-syntactic modifications that need to be done to the existing text in oder to go with the modifications? How to compute all this efficiently? An idea towards computing them efficiently could be to make use of RTGen's chart datastructure. It could serve as a pool of pre-computed intermediate alternative realisations which could be selectively picked up for combination in different contexts. Or keeping a dynamically modified generation forest from which the most appropriate combination of lexical items is evaluated and extracted.

5.2.3 Using GramEx

The *GramEx* framework is being integrated into two systems. The first one, called I-FLEG[43] (Interactive French Learning Game, [Amoia *et al.*, 2012; Denis *et al.*, 2012]), is a serious game prototype designed to explore the potential contribution of virtual environments in conjunction with NLP technology to ICALL. To learn French with I-FLEG, the learner moves her avatar inside a virtual 3D house and clicks on objects thereby triggering the display of language learning exercices. Altogether, they are meant to support a free (e.g. the learner explores the virtual world and decides on the type of (grammar) activities she wants to solve), situated (e.g. the content of the interactions e.g. messages from the system or activities are related to the learner current position in the virtual environment) and interactive (e.g. there is, yet basic, system initiative communication and feedback –score and correct answers–) learning environment. Briefly, the semantic information describing facts about the 3D world (i.e., the ontology describing the 3D objects) is exploited for the generation of sentences and grammar exercises. Given the input specification of the grammar point to be exercised and the object touched by the learner, *GramEx* produces an exercise item which allows to exercise the grammar point and that verbalises facts about the touched object.

The second is a web workbook called W-FLEG[44], where the learner can select a practice activity (teaching goal plus exercise type). Exercise items are provided for the select practice and feedback is also provided in the form of scores and correct

[43] `http://talc.loria.fr/I-FLEG.html`
[44] `http://talc.loria.fr/W-FLEG.html`

answers.

Data collection. Both I-FLEG and W-FLEG are web-based and rely on a database for permanent storage of learner interactions. Thus, this facilitates their use to collect data: learner's written language as well as extra information about the learner interactions. This additional information consists of (i) the teaching goal and exercise type, (ii) the exercise item question and expected answers, (iii) the score, (iv) the time spent on solving it, and (v) summary information that can be derived. For instance, how many times did she try the same exercise item until it got it right. This type of data could provide a rich resource to be exploited for research in second language acquisition and language teaching in different ways (cf. [Pravec, 2002] and references therein). For instance, the analysis of learner errors or the optimisation of instructional models as proposed by Pietquin *et al.* (2011).

Teacher mode and content edition. An important aspect towards putting *GramEx* to use is to supply an interface where language instructors could define exercices and edit content. For instance, either for I-FLEG's house ontology (or any other domain, e.g. restaurant or greengrocer's shop) or, in general, for any content or topic lesson for W-FLEG, minimal content needs to be edited. There are different ways to provide this functionality without requiring the teacher to be specialist in formal linguistic or semantic representation methods. Possible options include entering content using some kind of template-based interface or parsing simple sentences. Yet another possibility is to use a NLG based knowledge editor ([Scott *et al.*, 1998; Hallett *et al.*, 2007; Power, 2012]). In this editor, the author composes natural language text, while transparently for her, editing an underlying knowledge representation.

Evaluation. The integration of *GramEx* into a 3D environment provides a grammar-focused approach embedded in meaningful interactions, i.e. the content of the exercises "talks" about the virtual environment. Thus, it provides a setting in which to carry out comparative evaluations along the lines of Nagata (1996) to assess, for instance, the effectiveness of a contextualised setting to learn lexicon, morphology and syntax ([Macwhinney, 1995]).

5.2.4 Beyond grammar exercises

NLG has been used in *GramEx* for the generation of lexico-syntactic controlled grammar exercises. More precisely, *GramEx* supports the specification of pedagogical goals

in terms of lexico-syntactic features and the type of exercise activity (i.e. Fill-in-the-blank, Suffle or sentence transformation). However, the possibility to control how a given content is verbalised, and in particular, the paraphrase generation and selection mechanisms of the *SemTAG*-based generator, could be exploited in other ways for language learning. For instance, in a game or interactive task embedded in a virtual environment (e.g. [Denis *et al.*, 2010]), instructions could be adapted in syntax to the learner language skills as well as reformulations used to make communication effective or provide reinforcement feedback. In a dialogue setting, reformulations could be used to correct learner utterances or to rephrase some content not understood by the learner. In these cases, the set of constraints for generation (e.g. pedagogical goals) would be determined automatically (based on some knowledge) rather than predefined in the grammar exercise specifications. The parameters manipulated in those constraints could involve: learner language skills, the task or activity and the teaching goals.

Towards the creation of situated learning activities where emphasis is placed on communication rather than focused on grammar, a necessary extension would be to couple generated sentences with communicative goals (e.g. [Zock and Quint, 2004]). This could be done either by integrating into *SemTAG* a functional dimension describing the functional alternatives associated to syntactic forms ([Halliday, 1985; McCoy *et al.*, 1992]) or by using machine learning techniques to associate communicative goals with *SemTAG* generated sentences.

Appendices

Appendix A

GramEx pedagogical goals and exercise items

A.1 Excerpt of pedagogical goals

In what follows, we give an extract of the pedagogical goals defined with *GramEx* in Section 4.3.1. For each of them, we give a sort of header with its definition, i.e., how they are defined in *GramEx*, then, we give an excerpt of the concrete exercise items generated by *GramEx*. Each pedagogical goal is defined by a linguistic phenomena to be practiced and an activity type. The field **Pedagogical goal** gives the name of the pedagogical goal. The field **GramEx query (stem)** gives the syntactic and morpho-syntactic specification written in the *GramEx* boolean constraints language (cf. Section 4.2.2). This specification serves to retrieve sentences and might describe the linguistic phenomena to practice as well as other (morpho-)syntactic constraints desired for the teaching goal (cf. Sections 4.1 and 4.2.2). The field **Activity type** indicates the type of exercise that will be built from the selected sentences. In Table A.1, we list exercise types defined in *GramEx*. Finally, in some cases, depending on the type of exercise, additional configuration is specified. For instance, in the case of Fill-in-the-blank exercises the field **Config FIB_LexicalFeatures** specifies the conditions to identify the blank to build the exercise question. Indeed, this field specifies a set of (morpho-)syntactic constraints in the form of a feature structure. These (morpho-)syntactic constraints are defined over and evaluated against the linguistic information that is associated by the generation process with the output sentences (cf. Section 4.2.1).

In Table A.1 we detail all the exercise types that are available in *GramEx*. As discussed in Section 4.3 *GramEx* supports three different types of FIB questions

Description	activityType
Syntax Scramble	SYNTAX_SCRAMBLE
Fill in the blank -morphology on a given lemma	MORPHOLOGY_FIB
Transformation	TRANSFORMATION
Fill in the blank -missing word	SYNTAX_FIB
Morphology Scramble	MORPHOLOGY_SCRAMBLE
Fill in the blank -from a set of hint features	SYNTAX_FIB_FEATURES
Fill in the blank -from a given lemma and a set of hints	MORPHOLOGY_FIB_FEATURES

Table A.1: List of activity types implemented in *GramEx*.

which correspond with some of the activity types listed in Table A.1. All Fill-in-the-blank questions remove a word from the selected sentence, but *GramEx* provides three different ways of building the final Fill-in-the-blank exercise. One consists in replacing the chosen word by a blank (FIBBLNK), this corresponds to the SYNTAX_FIB activity type, the other replaces the chosen word by a lemma (FIBLEM), implemented by the MORPHOLOGY_FIB activity type, and the last one consists in replacing the word by a set of features used to help the learner guess the solution (FIBHINT), this corresponds to the SYNTAX_FIB_FEATURES activity type.

The Shuffle questions (MSHUF) of Section 4.3 correspond to the SYNTAX_SCRAMBLE and MORPHOLOGY_SCRAMBLE activity types. The difference between them is whether the words or the lemmas of the original sentence are shuffled.

The TRANSFORMATION activity type corresponds to the transformation exercises discussed in Section 4.4. This activity type is further specialised into ACTIVE_PASIVE, INTERROGATIVE_ADJECTIVE, INTERROGATIVE_PRONOUN, PERSONAL_PRONOUN, YNQUESTION_EST_CE_QUE_NEST_CE_PAS. They correspond to the transformation-based exercises Active/Passive, Wh-Questions, Pronominalisation and YN-Questions, respectively, of Section 4.4.

Pedagogical goal (15): Preposition - Fill in the blank -missing word
Activity type: SYNTAX_FIB
Config FIB_LexicalFeatures: [cat=p]
GramEx query (stem): $((?\text{PREP\_S\_ARG\_SUBCAT} \wedge (cat : p \vee cat : c)) \vee (?\text{PREP\_N\_ARG\_SUBCAT} \wedge cat : p) \vee (?\text{PREP\_MOD} \wedge cat : p))$

```
(Q)  les chemises propres sont sorties .........  Tex du lave linge
(S) par

(Q)  c' est Tex qui verse du lait .........  le bol
(S) dans

(Q)  les assiettes sont .........  le placard
(S) dans
```

```
(Q)  Tex prête la bouilloire .........  Tom
(S) à

(Q)  la pendule est accrochée .........  Tex au  mur
(S) par

(Q)  le réfrigérateur est montré à Tom .........  Tex
(S) par

(Q)  c' est Tex qui montre la cuisinière .........  Tom
(S) à

(Q)  c' est Tex qui s' occupe .........  la plante qui meurt
(S) de

(Q)  les chemises propres sont sorties .........  Tex du lave linge
(S) par

(Q)  Tex s' assied .........  la chaise
(S) sur
```

Pedagogical goal (22): Noun number: singular and plural - Fill in the blank -morphology on a given lemma

Activity type: `MORPHOLOGY_FIB`

Config FIB_LexicalFeatures: `[cat=n; nomPropre=-]`

GramEx query (stem): [noun $\wedge$ $num : pl$] $\wedge$?BASIC_CLAUSE_1FV

```
(Q)  c' est Tex qui a lavé les ......... (table)
(S) tables

(Q)  Tex adore les ......... (bibliothèque) anglaises
(S) bibliothèques

(Q)  c' est Tex qui sortit des ......... (livre) de la bibliothèque
(S) livres

(Q)  c' est Tex qui a éteint les ......... (lampe)
(S) lampes

(Q)  Tex range les ......... (tasse) dans le placard
(S) tasses

(Q)  ce sont les ......... (assiette) blanches que j' aime beaucoup
(S) assiettes

(Q)  Tex adore les vieux ......... (réfrigérateur)
(S) réfrigérateurs

(Q)  c' est Tex qui a arrosé les ......... ( plante )
```

```
(S) plantes

(Q)  ce sont des ......... (livre) que Tex sortit de la bibliothèque
(S) livres

(Q)  Tex enlève les ......... (tasse) de la table
(S) tasses
```

Pedagogical goal (79): Noun gender: masculine and feminine - Fill in the blank - from a set of hint features

Activity type: SYNTAX_FIB_FEATURES

Config FIB_LexicalFeatures: [cat=d; def = +-; num=sg]|

GramEx query (stem): [noun $\wedge$ $det : -$] $\wedge$ [$cat : d \wedge wh : - \wedge \neg$possessiveDetSem] $\wedge$?BASIC_CLAUSE_1FV

```
(Q)  c' est Tex qui alluma ......... (article défini) radio
(S) la

(Q)  c' est ......... (article défini) lampe que Tex montre à Tom
(S) la

(Q)  il arrose ......... (article défini) plante
(S) la

(Q)  les assiettes sont sur ......... (article défini) table
(S) la

(Q)  c' est Tex qui poussait ......... (article défini) chaise
(S) la

(Q)  c' est ......... (article défini) réfrigérateur que Tex ouvrit
(S) le

(Q)  ce sont les assiettes qui sont sur ......... (article défini) table
(S) la

(Q)  Tex sort des livres de ......... (article défini) bibliothèque
(S) la

(Q)  c' est Tex qui rangea ......... (article défini) placard
(S) le

(Q)  c' est Tex qui décroche ......... (article défini) téléphone
(S) le
```

Pedagogical goal (14): Adjective morphology - Fill in the blank -morphology on a given lemma

Activity type: `MORPHOLOGY_FIB`
Config FIB_LexicalFeatures: `[cat=adj]`
GramEx query (stem): $[cat : adj]$

```
(Q)  est-ce que la plante ......... (vert) est arrosée par Tex ?
(S) verte

(Q)  les ......... (petit) poêles sont préférées par Tex
(S) petites

(Q)  la lampe est ......... (moderne)
(S) moderne

(Q)  est ce que Tex essuie les ......... (petit) cuillères
(S) petites

(Q)  les bouilloires sont ......... (rouge)
(S) rouges

(Q)  est ce qu' il est ......... (blanc)
(S) blanc

(Q)  la pizza est ......... (délicieux)
(S) délicieuse

(Q)  le lave vaisselle est ......... (vide)
(S) vide

(Q)  Tex déteste des machines à café ......... (italien)
(S) italiennes

(Q)  est-ce que les ......... (petit) cuillères sont essuyées par Tex ?
(S) petites
```

Pedagogical goal (23): Adjectives that precede the noun - Fill in the blank -morphology on a given lemma
Activity type: `MORPHOLOGY_FIB`
Config FIB_LexicalFeatures: `[cat=adj]`
GramEx query (stem): (EpithAnte $\vee$ n0vAAnte)

```
(Q) Tex range les ......... (petit) tasses
(S) petites

(Q) Tex aime les ......... (vieux) radios
(S) vieilles

(Q) Tex adore les ......... (grand) tables
(S) grandes
```

```
(Q) la petite assiette qui est ......... (joli) est blanche
(S) jolie

(Q) la jolie assiette qui est ......... (petit) est blanche
(S) petite

(Q) Tex préfère les ......... (ancien) machines à café
(S) anciennes

(Q) le petit évier qui est propre est ......... (blanc)
(S) blanc

(Q) c' est Tex qui adore les ......... (vieux) réfrigérateurs
(S) vieux

(Q) c' est Tex qui allumera le ......... (petit) radiateur
(S) petit

(Q) c' est Tex qui déteste les ......... (petit) pendules
(S) petites
```

Pedagogical goal (54): Attributive adjectives - Syntax Scramble

Activity type: `SYNTAX_SCRAMBLE`

GramEx query (stem): (EpithAnte $\vee$ EpithPost)

```
(Q) petite / la / lampe / Tex / préfère
(S) Tex préfère la petite lampe

(Q) Tex / adore / grandes / tables / les
(S) Tex adore les grandes tables

(Q) ouvre / le / Tex / linge / vieux / lave
(S) Tex ouvre le vieux lave linge

(Q) les / range / tasses / Tex / petites
(S) Tex range les petites tasses

(Q) ? / t-il / arrose / la / plante / Tex / verte
(S) Tex arrose t-il la plante verte ?

(Q) adore / Tex / bouilloire / anglaise / la
(S) Tex adore la bouilloire anglaise

(Q) aime / blanches / les / assiettes / j' / beaucoup
(S) j'aime beaucoup les assiettes blanches

(Q) bibliothèques / anglaises / Tex / adore / les
(S) Tex adore les bibliothèques anglaises

(Q) beaucoup / éviers / aime / j' / les / blancs
(S) j'aime beaucoup les éviers blancs
```

```
(Q) pizza / la / brûle / cuit / qui / petite
(S) la petite pizza qui cuit brûle
```

Pedagogical goal (24): Irregular adjectives - Fill in the blank -morphology on a given lemma
Activity type: `MORPHOLOGY_FIB`
Config FIB_LexicalFeatures: `[cat=adj; flexion=irreg]`
GramEx query (stem): $[cat : adj \wedge flexion : irreg]$

```
(Q)  Tammy a une voix ......... ( doux )
(S)  [douce]

(Q)  Tammy avait une voix ......... ( doux )
(S)  [douce]

(Q)  c' est une voix ......... ( doux ) que Tammy avait
(S)  [douce]

(Q)  c' est Tammy qui avait une voix ......... ( doux )
(S)  [douce]
```

Pedagogical goal (24): Participles as adjectives - Fill in the blank -morphology on a given lemma
Activity type: `MORPHOLOGY_FIB`
Config FIB_LexicalFeatures: `[cat=v; mode=ppart/ppst]`
GramEx query (stem): $[cat : v \wedge (mode : ppst \vee mode : ppart) \wedge$ n0vAPredicative$]$

```
(Q)  Tammy est ......... ( épuisant )
(S)  épuisante

(Q)  c' est Tammy qui est ......... ( épuisant )
(S)  épuisante

(Q)  Tammy est ......... ( épuisé )
(S)  épuisée

(Q)  c' est Tammy qui est ......... ( épuisé )
(S)  épuisée
```

Pedagogical goal (28): Prepositions with infinitives - Fill in the blank -missing word
Activity type: `SYNTAX_FIB`
GramEx query (stem): ¬AbstractModifier∧[?PREP_INF_COMPL]∧?BASIC_CLAUSE_1FV

```
(Q)  il a commencé .........  écrire un livre
(S)  à

(Q)  il oublie .........  fermer la bibliothèque
(S)  de

(Q)  il vient .........  réparer la machine à café
(S)  de

(Q)  c' est lui qui a commencé .........  écrire un livre
(S)  à

(Q)  il vient .........  allumer les radiateurs
(S)  d'

(Q)  il a refusé .........   répondre  au téléphone
(S)  de

(Q)  tu essayes .........  éteindre la radio
(S)  d'

(Q)  il promet .........  réparer la chaise
(S)  de
```

Pedagogical goal (29): Subject pronouns - Fill in the blank - from a set of hint features
Activity type: `SYNTAX_FIB_FEATURES`
Config FIB_LexicalFeatures: `[cat=cl; func=suj; refl= -]`
GramEx query (stem): [pronounQuantifierSem $\wedge$ *cat* : *cl*]

```
(Q)  c' est le bol que ......... (gen:m; num:sg; pers:1) prends
(S) je

(Q)  ce sont les plantes qu' ......... (gen:m; num:sg; pers:3) arrose
(S) il

(Q)  ......... (pers:3) est la plante que Tex lui donne
(S) c'

(Q)  ......... (gen:f; num:sg; pers:3) est arrosée par Tex
(S) elle

(Q)  ......... (pers:3) est Tex qui la range
(S) c'

(Q)  ......... (gen:m; num:sg; pers:3) vient de fermer le four à micro-ondes
(S) il

(Q)  ......... (gen:m; num:sg; pers:3) range les tasses
(S) il
```

```
(Q)  ......... (pers:3) est la pizza qu' il fait
(S) c'

(Q)  ......... (pers:3) est le téléphone que je décroche
(S) c'

(Q)  ......... (gen:m; num:sg; pers:3) s' occupe d' une plante
(S) il
```

Pedagogical goal (52): Adjective order - Syntax Scramble
Activity type: SYNTAX_SCRAMBLE
GramEx query (stem): $[cat : adj]$

```
(Q) assiette / ? / est / blanche / quelle
(S) quelle assiette est blanche ?

(Q) rouge / la / bouilloire / est / qui
(S) la bouilloire qui est rouge

(Q) ? / elle / est / blanche
(S) elle est blanche ?

(Q) grandes / Tex / aime / les / assiettes
(S) Tex aime les grandes assiettes

(Q) ? / la / plante / Tex / verte / arrose / t-il
(S) Tex arrose t-il la plante verte ?

(Q) blanche / elle / t-elle / est / ?
(S) elle est t-elle blanche ?

(Q) adore / bouilloire / que / Tex / anglaise / la
(S) la bouilloire anglaise que Tex adore

(Q) rouge / la / bouilloire / est / qui
(S) la bouilloire qui est rouge

(Q) bruyants / sont / réfrigérateurs / les
(S) les réfrigérateurs sont bruyants

(Q) beaucoup / aime / blanches / assiettes / j' / les
(S) j'aime beaucoup les assiettes blanches
```

Pedagogical goal (56): Verb conjugation: simple past - fill in the blank -morphology on a give lemma and a set of hints
Activity type: MORPHOLOGY_FIB_FEATURES

Config FIB_LexicalFeatures: `[cat=v;mode=ind;tense=past]`
GramEx query (stem): $[cat : v \ \wedge \ tense : past \ \wedge \ mode : ind]$

```
(Q)  c' est Tex qui ......... (éteindre; tense:simple past; mood:indicative) la lampe
(S) éteignit

(Q)  c' est la bouilloire que Tex ......... (ranger; tense:simple past; mood:indicative)
(S) rangea

(Q)  Tex ......... (vider; tense:simple past; mood:indicative) le lave linge
(S) vida

(Q)  c' est la bouilloire que Tex ......... (remplir; tense:simple past; mood:indicative)
(S) remplit

(Q)  c' est la table que Tex ......... (mettre; tense:simple past; mood:indicative)
(S) mit

(Q)  c' est Tex qui ......... (éteindre; tense:simple past; mood:indicative) la lampe
(S) éteignit

(Q)  ce sont les tables que Tammy ......... (décorer; tense:simple past; mood:indicative)
(S) décora

(Q)  le four à micro-ondes ......... (etre; tense:simple past; mood:indicative) ouvert par Tex
(S) fut

(Q)  c' est le réfrigérateur que Tex ......... (laver; tense:simple past; mood:indicative)
(S) lava

(Q)  le téléphone ......... (etre; tense:simple past; mood:indicative) décroché par Tex
(S) fut
```

Pedagogical goal (57): Verb conjugation: simple future - fill in the blank -morphology on a give lemma and a set of hints
Activity type: `MORPHOLOGY_FIB_FEATURES`
Config FIB_LexicalFeatures: `[cat=v;mode=ind;tense=fut]`
GramEx query (stem): $[cat : v \ \wedge \ tense : fut \ \wedge \ mode : ind]$

```
(Q)  c' est la radio que Tex ......... (allumer; tense:future; mood:indicative)
(S) allumera

(Q)  tu ......... (mettre; tense:future; mood:indicative) la table
(S) mettras

(Q)  c' est la table que Tex ......... (laver; tense:future; mood:indicative)
(S) lavera

(Q)  c' est Tex qui ......... (pousser; tense:future; mood:indicative) la chaise
```

```
(S) poussera

(Q)  c' est Tex qui ......... (ranger; tense:future; mood:indicative) le placard
(S) rangera

(Q)  c' est la table que Tex ......... (mettre; tense:future; mood:indicative)
(S) mettra

(Q)  des livres ......... (etre; tense:future; mood:indicative) sortis de la bibliothèque par Tex
(S) seront

(Q)  la lampe ......... (etre; tense:future; mood:indicative) allumée par Tex
(S) sera

(Q)  Tex ......... (nettoyer; tense:future; mood:indicative) la poêle
(S) nettoiera

(Q)  c' est Tex qui ......... (décrocher; tense:future; mood:indicative) le téléphone
(S) décrochera
```

Pedagogical goal (37): -ir verbs in present - Fill in the blank -morphology on a given lemma

Activity type: `MORPHOLOGY_FIB`

Config FIB_LexicalFeatures: `[cat=v; group=ir; lemanchor=none]`

GramEx query (stem): [VerbSem $\wedge$ $cat : v \wedge tense : pres \wedge group : ir \wedge mode : ind$]

```
(Q)  c' est lui qui ......... (venir) de réparer la machine à café
(S)  vient

(Q)  Tex ......... (choisir) les petites poêles
(S)  choisit

(Q)  ce sont les petites poêles que Tex ......... (choisir)
(S)  choisit

(Q)  c' est lui qui ......... (venir) de fermer le four à micro-ondes
(S)  vient
```

Pedagogical goal (71): Pronominal verbs - Syntax Scramble

Activity type: `SYNTAX_SCRAMBLE`

Config FIB_LexicalFeatures:

`[cat=v; pronominal= +; mode=ind; tense=pres; aspect= indet; lemanchor=none]`

GramEx query (stem): [$cat : v \wedge pronominal : + \wedge mode : ind \wedge tense : pres \wedge aspect : indet$]

```
(Q)  Tex / s' / chaise / la / sur / assied
```

```
(S)  Tex s'assied sur la chaise

(Q)  s' / pizzas / Tex / intéresse / aux
(S)  Tex s'intéresse aux pizzas

(Q)  plante / est / lui / une / d' / s' / occupe / c' / qui
(S)  c' est lui qui s'occupe d'une plante

(Q)  s' / c' / pizzas / intéresse / Tex / est / aux / qui
(S)  c' est Tex qui s'intéresse aux pizzas

(Q) Tex / pizza / intéresse / la / s' / à
(S)  Tex s'intéresse à la pizza

(Q)  en / Tex / occupe / s'
(S)  Tex s'en occupe

(Q)  en / qui / c' / Tex / occupe / s' / est
(S)  c' est Tex qui s'en occupe

(Q)  occupe / s' / il / en
(S)  il s'en occupe

(Q)  la  / appelle / tatou / Tammy / s'
(S)  la tatou s'appelle Tammy

(Q)  occupe / des / s' / plantes / Tex
(S)  Tex s'occupe des plantes
```

Pedagogical goal (81): verb conjugation: present[45] - fill in the blank -morphology on a give lemma and a set of hints

Activity type: `MORPHOLOGY_FIB_FEATURES`

Config FIB_LexicalFeatures: `[cat=v;mode=ind;tense=pres]`

GramEx query (stem): $[cat : v \ \wedge \ tense : pres \ \wedge \ mode : ind]$

```
(Q)  les petites cuillères ......... (etre; tense:present; mood:indicative) t-elles essuyées par Tex
(S) sont

(Q)  les lave vaisselle ......... (avoir; tense:present; mood:indicative) été vidés par Tex
(S) ont

(Q)  c' est Tex qui s' en ......... (occuper; tense:present; mood:indicative)
(S) occupe

(Q)  Tex ......... (adorer; tense:present; mood:indicative) les grandes tables
(S) adore
```

[45] Here we provide the "verb conjugation: present" instead of the subjunctive one. The definition of the Pedagogical goal "verb conjugation: subjunctive" differ only in the Config FIB_LexicalFeatures and GramEx query (stem) in that it has the mode=subj speficifation rather than mode=pres.

```
(Q)  c' est lui qui ......... (promettre; tense:present; mood:indicative) de réparer la chaise
(S) promet

(Q)  c' ......... (etre; tense:present; mood:indicative) la chaise que Tex poussait
(S) est

(Q)  ce sont les tasses que Tex ......... (avoir; tense:present; mood:indicative) rangées
(S) a

(Q)  c' est lui qui ......... (avoir; tense:present; mood:indicative) oublié de débarasser la table
(S) a

(Q)  Tex rêve de la pizza qui ......... (bruler; tense:present; mood:indicative)
(S) brule

(Q)  les vieux réfrigérateurs ......... (etre; tense:present; mood:indicative) adorés par Tex
(S) sont
```

The figures below show examples of interactions within I-FLEG. The exercises are generated by *GramEx* and correspond to some of the teaching goals previously detailed. Figure A.1 show a question of the type "Preposition - Fill in the blank -missing word" given to the learner and Figure A.2 the answer entered by her with the feedback returned by I-FLEG. The other pair of figures, namely Figure A.3 and Figure A.4 show an interaction for the *Adjective order - Syntax Scramble* pedagogical goal. In the first case, the learner enters an incorrect answer and therefore I-FLEG shows the expected correct answer. In the second case, the answer entered by the learner is correct, then I-FLEG gives a positive message that confirms this.

A.2 Excerpt of transformation-based grammar exercices

In what follows, we give an excerpt of the exercises generated by *GramEx* for the pedagogical goals defined in Section 4.4 concerning Reformulation type of exercises.

Pedagogical goal (19): Passive voice - Transformation
Activity type: `TRANSFORMATION`
Config TASType: `ACTIVE_PASIVE`
GramEx query (stem): [activeVerbMorphology $\wedge$ (binaryRel $\vee$ ternaryRel) $\wedge$ SubjectSem $\wedge$ ObjectSem $\wedge$ $tense : pres$]

```
(Q)  il les adore
(S)  ils sont adorés par lui
```

Figure A.1: An example of exercise of the "(15) Preposition - Fill in the blank -missing word" pedagogical goal given to the learner.

Figure A.2: Answer entered by the learner and feedback given by I-FLEG to the learner for the preposition exercise question in Figure A.1.

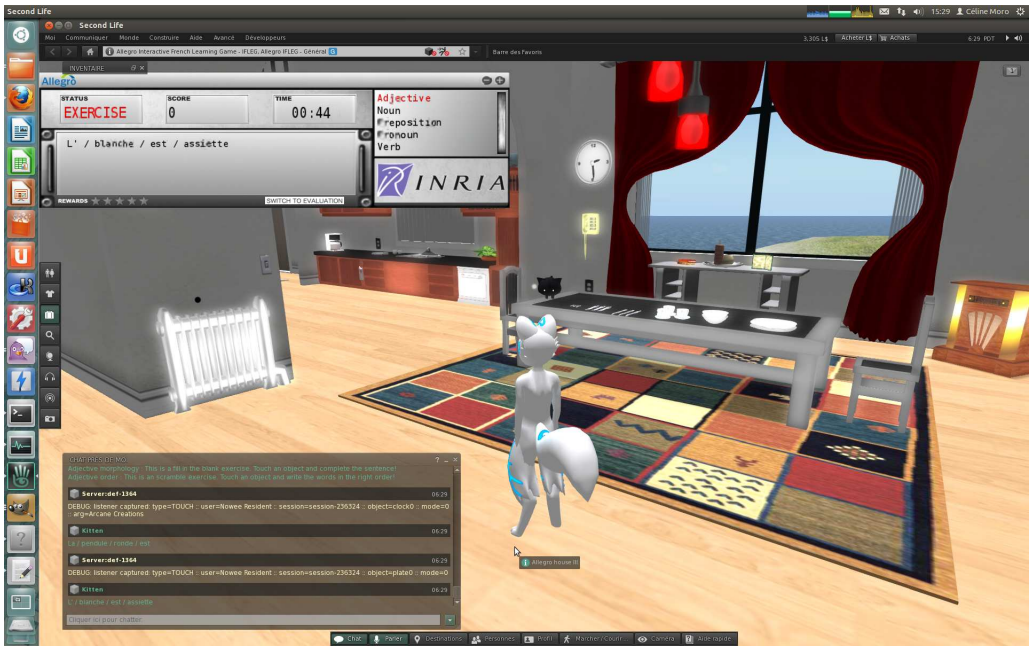


Figure A.3: An example of exercise of the "(52) Adjective order - Syntax Scramble" pedagogical goal given to the learner.

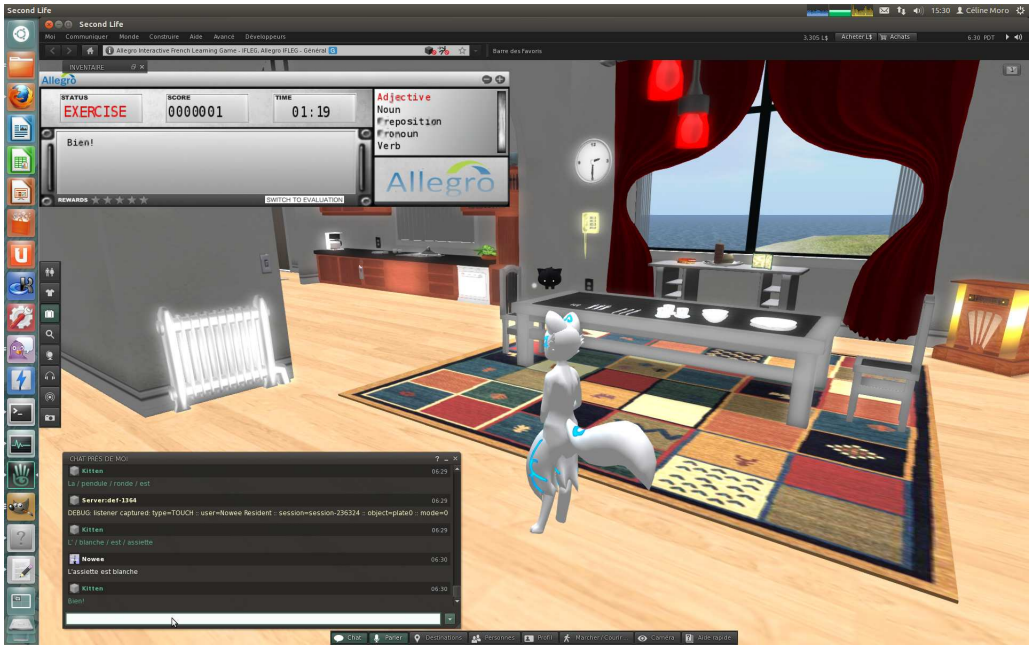


Figure A.4: Answer entered by the learner and feedback given by I-FLEG to the learner for the adjectives exercise question in Figure A.3.

```
(Q)  c' est lui que Tammy adore
(S)  c' est lui qui est adorée par Tammy
(S)  c' est par Tammy qu' il  est adoré

(Q)  Tex  présente  Tammy à Tom
(S)  Tammy est présentée par Tex à Tom
(S)  Tammy est présentée à Tom par Tex

(Q)  qui présente Tammy à Tom  ?
(S)  par qui Tammy est présentée à Tom  ?

(Q)  Tex a fait la tarte
(S)  la tarte a été faite par Tex

(Q)  il adore la petite tatou
(S)  la petite tatou est adorée par lui

(Q)  c' est lui que Tex aime beaucoup
(S)  c' est lui qui est aimé beaucoup par Tex

(Q)  Tammy adore t-elle la petite tatou ?
(S)  la  petite tatou est  t-elle  adorée par Tammy ?

(Q)  est ce que c' est Tammy qui adore la petite tatou ?
(S)  est ce que c' est par Tammy que la petite tatou est adorée ?

(Q)  Tammy sait que Tom a fait la tarte
(S)  Tammy sait que la tarte a été faite par Tom

(Q)  c' est Tammy qui sait que Tom a fait la tarte
(S)  c' est Tammy qui sait que la tarte a été faite par Tom
```

Pedagogical goal (44): Personal pronouns - Transformation

Activity type: `TRANSFORMATION`

Config TASType: `PERSONAL_PRONOUN`

GramEx query (stem): (propername $\vee$ noun) $\wedge$ [$tense : pres \wedge mode : ind$]

```
(Q)  c' est lui que Tammy adore
(S)  c' est lui  que  elle  adore

(Q)  Tammy  l' adore
(S)  elle  l' adore

(Q)  c' est lui qui adore la tatou
(S)  c' est lui  qui  l' adore

(Q)  Tammy parle avec Tex
(S)  Tammy parle avec lui

(Q)  c' est Tammy qui parle avec Tex
(S)  c' est Tammy qui parle avec lui
```

```
(Q)  adore t-il la petite tatou ?
(S)  l' adore t-il ?

(Q)  Tex aime beaucoup le café
(S)  il aime beaucoup le café

(Q)  quelle petite tatou dort  ?
(S)  qui dort  ?

(Q)  la tatou dort
(S)  elle dort

(Q)  la petite tatou qui chante dort
(S)  elle dort

(Q)  Tammy dort
(S)  elle dort

(Q)  c' est Tammy qui dort n' est ce pas ?
(S)  c' est lui qui dort n' est ce pas ?
```

Pedagogical goal (45): yes/no questions (est-ce que or n'est-ce pas) - Transformation

Activity type: `TRANSFORMATION`

Config TASType: `YNQUESTION_EST_CE_QUE_NEST_CE_PAS`

GramEx query (stem): $\neg$(questionSem $\vee$ questionmarkSem $\vee$ whPronoun $\vee$ questionCliticSem $\vee$ UnboundedQuestion $\vee$ qtilSem) $\wedge$ [$tense : pres \wedge mode : ind$]

```
(Q)  c' est lui qu' elles  adorent
(S)  est ce que c' est lui qu' elles  adorent ?

(Q)  il l' adore
(S)  il l' adore  ?

(Q)  il l' adore
(S)  est-ce qu' il l' adore ?

(Q)  il l' adore
(S)  il l' adore  n' est ce pas ?

(Q)  la tatou dort beaucoup
(S)  est-ce que la tatou dort beaucoup ?

(Q)  Tammy parle avec Tex
(S)  Tammy parle avec Tex n' est ce pas ?

(Q)  Tammy leur sourit
(S)  est-ce que Tammy leur sourit ?

(Q)  c' est la petite tatou qui chanta qui dort
(S)  est-ce que c' est la petite tatou qui chante qui dort ?
```

```
(Q)  elle dort
(S)  est-ce qu' elle dort ?
```

Pedagogical goal (46): Interrogative pronouns - Transformation

Activity type: `TRANSFORMATION`

Config TASType: `INTERROGATIVE_PRONOUN`

GramEx query (stem): ¬(questionSem∨questionmarkSem∨whPronoun∨questionCliticSem∨ UnboundedQuestion ∨ qtilSem) ∧ [*tense* : *pres* ∧ VerbSem]

```
(Q)  Tammy est présentée à Tom par Tex
(S)  à qui Tammy est présentée par Tex  ?

(Q)  ma tatou dort
(S)  qui dort  ?

(Q)  elles sont adorées par Tammy
(S)  qui est adoré par Tammy  ?

(Q)  Tammy leur sourit
(S)  qui leur sourit  ?

(Q)  Tammy adore le petit tatou
(S)  qui Tammy adore  ?

(Q)  la petite tatou qui chante dort
(S)  qui dort  ?

(Q)  Tammy sourit â Tex
(S)  à qui sourit Tammy  ?

(Q)  elle dort
(S)  qui dort  ?

(Q)  la petite tatou est adorée par Tammy
(S)  qui est adoré par Tammy  ?
```

Pedagogical goal (48): interrogative adjective quel - Transformation

Activity type: `TRANSFORMATION`

Config TASType: `INTERROGATIVE_ADJECTIVE`

GramEx query (stem): ¬(questionSem ∨ questionmarkSem ∨ whPronoun ∨ questionCliticSem ∨ UnboundedQuestion ∨ qtilSem) ∧ *tense* : *pres* ∧ noun

```
(Q)  la petite tatou qui chante dort
(S)  quelle tatou dort  ?

(Q)  la petite tatou qui chante dort
(S)  quelle petite tatou dort  ?
```

```
(Q)  la tatou dort beaucoup
(S)  quelle tatou dort beaucoup  ?

(Q)  la petite tatou dort
(S)  quelle tatou dort  ?
```

Bibliography

[Abeillé and Rambow, 2000] Anne Abeillé and Owen Rambow. *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*, chapter Tree Adjoining Grammar: An Overview. Cambridge University Press, 2000.

[Alahverdzhieva, 2008] Katya Alahverdzhieva. XTAG using XMG. A Core Tree-Adjoining Grammar for English. Master's thesis, Universität des Saarlandes, Germany and Université Nancy 2, France, 2008.

[Aldabe *et al.*, 2006] Itziar Aldabe, Maddalen Lopez de Lacalle, Montse Maritxalar, Edurne Martinez, and Larraitz Uria. Arikiturri: an automatic question generator based on corpora and nlp techniques. In *Proceedings of the 8th international conference on Intelligent Tutoring Systems*, ITS'06, pages 584–594, Berlin, Heidelberg, 2006. Springer-Verlag.

[Amaral and Meurers, 2011] Luiz a. Amaral and Detmar Meurers. On using intelligent computer-assisted language learning in real-life foreign language teaching and learning. *ReCALL*, 23(1):4–24, January 2011.

[Amaral, 2011] Luiz Amaral. Revisiting current paradigms in computer assisted language learning research and development. *Ilha do Desterro*, 2011.

[Amoia *et al.*, 2012] Marilisa Amoia, Treveur Bretaudiere, Alexandre Denis, Claire Gardent, and Laura Perez-Beltrachini. A Serious Game for Second Language Acquisition in a Virtual Environment. *The Journal on Systemics, Cybernetics and Informatics (JSCI)*, pages 24–34, 2012.

[Bachman and Palmer, 1996] Lyle F. Bachman and Adrian S. Palmer. *Language Testing in Practice: Designing and Developing Useful Language Tests*. Oxford University Press, 1996.

[Bangalore and Joshi, 1999] Srinivas Bangalore and Aravind K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265, 1999.

[Bangalore and Rambow, 2000a] S. Bangalore and O. Rambow. Using TAGs, a tree model and a language model for generation. In *Proceedings of TAG+5*, Paris, France, 2000.

[Bangalore and Rambow, 2000b] Srinivas Bangalore and Owen Rambow. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, COLING '00, pages 42–48, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[Banik *et al.*, 2012] Eva Banik, Eric Kow, Vinay Chaudhri, Nikhil Dinesh, and Umangi Oza. Natural language generation for a smart biology textbook. In *Proceedings of the Seventh International Natural Language Generation Conference*, INLG '12, pages 125–127, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[Belz *et al.*, 2011] A. Belz, M. White, D. Espinosa, E. Kow, D. Hogan, and A. Stent. The first surface realisation shared task: Overview and evaluation results. In *Proc. of the 13th European Workshop on Natural Language Generation*, 2011.

[Biber and Conrad, 2010] Douglas Biber and Susan Conrad. Corpus linguistics and grammar teaching. Newsletter : Pearson Longman English Language Teaching, May 2010.

[Bick, 2005] Eckhard Bick. Grammar for Fun: IT-based Grammar Learning with VISL. In P. Juel, editor, *CALL for the Nordic Language*, pages 49–64, Copenhagen, 2005.

[Billot and Lang, 1989] Sylvie Billot and Bernard Lang. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, ACL '89, pages 143–151, Stroudsburg, PA, USA, 1989. Association for Computational Linguistics.

[Bodirsky *et al.*, 2005] Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. Well-nested drawings as models of syntactic structure. In *In Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*, pages 88–1. University Press, 2005.

[Bonfante *et al.*, 2004] Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *Proceedings of the 20th international conference on Computational*

Linguistics, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[Bos, 1995] Johan Bos. Predicate logic unplugged. In *In Proceedings of the 10th Amsterdam Colloquium*, pages 133–143, 1995.

[Bresnan *et al.*, 1982] Joan Bresnan, Ronald M. Kaplan, Stanley Peters, and Annie Zaenen. Cross-Serial Dependencies in Dutch. In *Linguistic Inquiry*, volume 13, pages 613–635. The MIT Press, 1982.

[Brew, 1992] Chris Brew. Letting the cat out of the bag. In *Proceedings of COLING*, 1992.

[Callaway, 2003] Charles B. Callaway. Evaluating coverage for large symbolic NLG grammars. In *18th IJCAI*, pages 811–817, Aug 2003.

[Candito and Kahane, 1998] Marie-Hélène Candito and Sylvain Kahane. Can the TAG derivation tree represent a semantic graph? An answer in the light of Meaning-Text Theory. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*, 1998.

[Carroll and Oepen, 2005] J. Carroll and S. Oepen. High efficiency realization for a wide-coverage unification grammar. *2nd IJCNLP*, 2005.

[Carroll *et al.*, 1999] John Carroll, Dan Flickinger, Ann Copestake, and Victor Poznański. An Efficient Chart Generator for (Semi-)Lexicalist Grammars. In *In Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99*, Toulouse, France, 1999.

[Chandrasekar and Srinivas, 1997] R. Chandrasekar and B. Srinivas. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, pages 183–190, 1997.

[Chao-Lin *et al.*, 2005] Liu Chao-Lin, Wang Chun-Hung, Gao Zhao-Ming, and Huang Shang-Ming. Applications of lexical information for algorithmically composing multiple-choice cloze items. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, EdAppsNLP 05, pages 1–8, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[Chen *et al.*, 1999] John Chen, Srinivas Bangalore, and K. Vijay-Shanker. New models for improving supertag disambiguation. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, EACL '99,

pages 188–195, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[Chen *et al.*, 2006] Chia-Yin Chen, Hsien-Chin Liou, and Jason S. Chang. Fast: an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, COLING-ACL '06, pages 1–4, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[Chomsky, 1957] N. Chomsky. *Syntactic Structures*. Mouton, 1957.

[Cohn and Lapata, 2009] T. Cohn and M. Lapata. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674, 2009.

[Comon *et al.*, 1997] H. Comon, M. Dauchet, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Technical report, 1997.

[Coniam, 1997] David Coniam. A preliminary inquiry into using corpus word frequency data in the automatic generation of english language cloze tests. *CALICO Journal*, 14:15–33, 1997.

[Copestake *et al.*, 2001] Ann Copestake, Alex Lascarides, and Dan Flickinger. An algebra for semantic construction in constraint-based grammars. In *In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001*, pages 132–139, 2001.

[Copestake *et al.*, 2005] A. Copestake, D. Flickinger, C. Pollard, and I.A. Sag. Minimal recursion semantics: An introduction. *Research on Language & Computation*, 3(2):281–332, 2005.

[Copestake, 2008] Ann Copestake. Dependency and (R)MRS. Technical report, 2008.

[Copestake, 2009] Ann Copestake. Slacker semantics: why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics, 2009.

[Crabbé, 2005] Benoit Crabbé. *Représentation informatique de grammaires d'arbres fortement lexicalisées : le cas de la grammaire d'arbres adjoints.* PhD thesis, Nancy University, 2005.

[Crabbé *et al.*, 2012] B. Crabbé, D. Duchier, C. Gardent, J. Leroux, and Y. Parmentier. XMG, eXtendible Meta Grammar. *Computational Linguistics*, 2012.

[De Groote, 2002] P. De Groote. Tree-adjoining grammars as abstract categorial grammars. In *Proceedings of TAG+6*, pages 145–150, 2002.

[Dechter, 1990] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, 1990.

[DeJong and Mooney, 1986] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine learning*, 1(2):145–176, 1986.

[Denis *et al.*, 2010] Alexandre Denis, Marilisa Amoia, Luciana Benotti, Laura Perez-Beltrachini, Claire Gardent, and Tarik Osswald. The GIVE-2 Nancy Generation Systems NA and NM. Technical report, GIVE challenge, 2010.

[Denis *et al.*, 2012] Alexandre Denis, Ingrid Falk, Claire Gardent, and Laura Perez-Beltrachini. Representation of linuguistic and domain knowledge for second language learning in virtual worlds. In *LREC 2012: Posters*, Istanbul, Turkey, May 2012.

[Duchier and Debusmann, 2001] Denys Duchier and Ralph Debusmann. Topological dependency trees: A constraint-based account of linear precedence. In *In Proceedings of the 39th ACL*, 2001.

[Earley, 1970] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February 1970.

[Elhadad *et al.*, 1997] Michael Elhadad, Jacques Robin, et al. Surge: a comprehensive plug-in syntactic realization component for text generation. *Computational Linguistics*, 99(4), 1997.

[Espinosa *et al.*, 2008] Dominic Espinosa, Michael White, and Dennis Mehay. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Franconi *et al.*, 2010] Enrico Franconi, Paolo Guagliardo, and Marco Trevisan. An intelligent query interface based on ontology navigation. In *Workshop on Visual Interfaces to the Social and Semantic Web, VISSW*. Citeseer, 2010.

[Gardent and Kallmeyer, 2003] C. Gardent and L. Kallmeyer. Semantic construction in Feature-Based TAG. In *10th EACL*, Budapest, Hungary, 2003.

[Gardent and Kow, 2005] Claire Gardent and Eric Kow. Generating and selecting grammatical paraphrases. In *In Proceedings of the 10th European Workshop on Natural Language Generation*, 2005.

[Gardent and Kow, 2006] Claire Gardent and Eric Kow. Three reasons to adopt TAG-based surface realisation. In *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms*, TAGRF '06, pages 97–102, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[Gardent and Kow, 2007] Claire Gardent and Eric Kow. A symbolic approach to near-deterministic surface realisation using tree adjoining grammar. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 328–335, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Gardent and Kruszewski, 2012] Claire Gardent and German Kruszewski. Generation for grammar engineering. In *11th International Conference on Natural Language Generation (ENLG)*, 2012.

[Gardent and Narayan, 2012] Claire Gardent and Shashi Narayan. Error mining on dependency trees. In *ACL*, pages 592–600, 2012.

[Gardent and Perez-Beltrachini, 2010] C. Gardent and L. Perez-Beltrachini. RTG based surface realisation for TAG. In *Proceedings of COLING*, Beijing, China, 2010.

[Gardent and Perez-Beltrachini, 2012] Claire Gardent and Laura Perez-Beltrachini. Using FB-LTAG Derivation Trees to Generate Transformation-Based Grammar Exercises. In *TAG+11: The 11th International Workshop on Tree Adjoining Grammars and Related Formalisms*, Paris, France, September 2012.

[Gardent and Thater, 2001] C. Gardent and S. Thater. Generating with a grammar based on tree descriptions: a constraint-based approach. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 212–219. Association for Computational Linguistics, 2001.

[Gardent *et al.*, 2010] Claire Gardent, Benjamin Gottesman, and Laura Perez-Beltrachini. Comparing the performance of two TAG-based surface realisers using

controlled grammar traversal. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 338–346. Association for Computational Linguistics, 2010.

[Gardent *et al.*, 2011a] C. Gardent, B. Gottesman, L. Perez-Beltrachini, et al. Using regular tree grammars to enhance sentence realisation. *Natural Language Engineering*, 17(2):185, 2011.

[Gardent *et al.*, 2011b] Claire Gardent, Yannick Parmentier, Guy Perrier, and Sylvain Schmitz. Lexical Disambiguation in LTAG using Left Context. In *Proceedings of the 5th Language and Technology Conference - LTC'11*, 2011.

[Gardent, 2008] Claire Gardent. Integrating a unification-based semantics in a large scale lexicalised tree adjoining grammar for french. In *COLING'08*, Manchester, UK, 2008.

[Gottesman, 2009] B. Gottesman. Generating examples. Master's thesis, Erasmus Mundus Master Language and Communication Technology, Saarbrucken/Nancy, 2009.

[Hallett *et al.*, 2007] Catalina Hallett, Donia Scott, and Richard Power. Composing questions through conceptual authoring. *Computational Linguistics*, 33(1):105–133, 2007.

[Halliday, 1985] Michael A. K. Halliday. *An introduction to functional grammar*. Edward Arnold Press, 1985.

[Harbusch and Kempen, 2010] Karin Harbusch and Gerard Kempen. Automatic online writing support for l2 learners of german through output monitoring by a natural-language paraphrase generator. *WorldCall: International Perspectives on Computer-Assisted Language Learning*, 5:128, 2010.

[Harbusch *et al.*, 2007] Karin Harbusch, Camiel Van Breugel, Ulrich Koch, and Gerard Kempen. Interactive sentence combining and paraphrasing in support of integrated writing and grammar instruction: a new application area for natural language sentence generators. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 65–68, 2007.

[Harbusch *et al.*, 2008a] Karin Harbusch, Gergana Itsova, Ulrich Koch, and Christine Kühner. The Sentence Fairy: a natural-language generation system to support children's essay writing. *Computer Assisted Language Learning*, 21(4):339–352, 2008.

[Harbusch *et al.*, 2008b] Karin Harbusch, Gerard Kempen, and Theo Vosse. A natural-language paraphrase generator for on-line monitoring and commenting incremental sentence construction by l2 learners of german. *Proceedings of the WorldCALL 2008: Bridging the World through Technology Enhanced Language Learning*, 2008.

[Harbusch *et al.*, 2009] Karin Harbusch, Gergana Itsova, Ulrich Koch, and CHRISTINE Kühner. Computing accurate grammatical feedback in a virtual writing conference for german-speaking elementary-school children: An approach based on natural language generation. *CALICO Journal*, 20:626–643, 2009.

[Harbusch *et al.*, 2012] Karin Harbusch, Christine Franz, and Ulrich Koch. The teacher mode of the sentence fairy system: How to create your own e-learning writing lessons for german elementary school pupils. *ICERI2012 Proceedings*, pages 112–122, 2012.

[Harris, 1957] Z.S. Harris. Co-occurrence and transformation in linguistic structure. *Language*, 33(3):283–340, 1957.

[Heilman and Eskenazi, 2007] Michael Heilman and Maxine Eskenazi. Application of automatic thesaurus extraction for computer generation of vocabulary questions. In *Proceedings of Speech and Language Technology in Education (SLaTE2007)*, pages 65–68, 2007.

[Heilman *et al.*, 2008] Michael Heilman, Le Zhao, Juan Pino, and Maxine Eskenazi. Retrieval of reading materials for vocabulary and reading practice. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 80–88, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Hockenmaier and Steedman, 2007] Julia Hockenmaier and Mark Steedman. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank, 2007.

[Hwa, 2000] Rebecca Hwa. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 45–52, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[Joshi and Schabes, 1997] A. Joshi and Y. Schabes. *Handbook of Formal Languages*, chapter Tree-Adjoining Grammars. Springer, 1997.

[Joshi *et al.*, 1975] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163, February 1975.

[Joshi, 1985] Aravind K. Joshi. How Much Context-Sensitivity is Necessary for Characterizing Structural Descriptions—Tree Adjoining Grammars. In L.; Dowty, D.; Karttunen and A. Zwicky, editors, *Natural Language Processing—Theoretical, Computational and Psychological Perspective*. Cambridge University Press, New York, NY, 1985. Originally presented in 1983.

[Joshi, 1987] Aravind K. Joshi. The relevance of tree adjoining grammar to generation. In Gerard Kempen, editor, *Natural Language Generation*, pages 233–252. Martinus Nijhoff Press, Dordrect, The Netherlands, 1987.

[Kallmeyer, 2013] Laura Kallmeyer. Linear context-free rewriting systems. *Language and Linguistics Compass*, 7(1):22–38, 2013.

[Kanazawa, 2007] M. Kanazawa. Parsing and generation as datalog queries. In *Proceedings of ACL*, pages 176–183, 2007.

[Kanazawa, 2011] MAKOTO Kanazawa. Parsing and generation as datalog query evaluation. *To appear*, 2011.

[Kaplan and Bresnan, 1981] Ronald M Kaplan and Joan Bresnan. *Lexical-Functional Grammar: A formal system for grammatical representation.* Massachusetts Institute of Technology, Center for Cognitive Science, 1981.

[Karamanis *et al.*, 2006] Nikiforos Karamanis, Le An Ha, and Ruslan Mitkov. Generating multiple-choice test items from medical text: A pilot study. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 111–113, Sydney, Australia, 2006.

[Kay, 1986] M Kay. Readings in natural language processing. chapter Algorithm schemata and data structures in syntactic processing, pages 35–70. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986.

[Kay, 1996] Martin Kay. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204, Morristown, NJ, USA, 1996.

[Kempen and Harbusch, 2002] Gerard Kempen and Karin Harbusch. Performance grammar: A declarative definition. *Language and Computers*, 45(1):148–162, 2002.

[Kern, 2006] Richard Kern. Perspectives on technology in learning and teaching languages. *TESOL Quarterly*, 2006.

[Koller and Hoffmann, 2010] A. Koller and J. Hoffmann. Waking up a sleeping rabbit: On natural-language generation with FF. In *Proceedings of the 20th ICAPS (Short Papers)*, Toronto, Canada, 2010.

[Koller and Stone, 2007] A. Koller and M. Stone. Sentence generation as planning. In *Proceedings of 45th Annual Meeting of the Association for Computational Linguistics*, Prague, 2007.

[Koller and Striegnitz, 2002] A. Koller and K. Striegnitz. Generation as dependency parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, 2002.

[Kow, 2007] Eric Kow. *Surface realisation: ambiguity and determinism*. PhD thesis, Université Henri Poincaré, 2007.

[Krashen, 1982] Stephen D. Krashen. *Principles and Practice in Second Language Acquisition*. Pergamon Press, 1982.

[Kuhlmann, 2007] Marco Kuhlmann. *Dependency Structures and Lexicalized Grammars*. PhD thesis, Saarland University, 2007.

[Langkilde-Geary, 2002] I. Langkilde-Geary. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the INLG*, 2002.

[Langkilde, 2000] Irene Langkilde. Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 170–177, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[Lee and Seneff, 2007] John Lee and Stephanie Seneff. Automatic generation of cloze items for prepositions. *Proceedings of Interspeech*, pages 2173–2176, 2007.

[Levin and Evans, 1995] Lori S. Levin and David A. Evans. Alice-chan: A case study in icall theory and practice. *Intelligent language tutors: Theory shaping technology*, pages 77–97, 1995.

[Levy and Hubbard, 2005] Mike Levy and Philip Hubbard. Why call CALL "CALL"? *Computer Assisted Language Learning*, 2005.

[Levy, 1997] Mike Levy. *CALL: context and conceptualisation.* Oxford University Press, 1997.

[Li and Topolewski, 2002] Rong-Chang Li and David Topolewski. ZIP & TERRY: a new attempt at designing language learning simulation. *Simulation and Gaming*, 33(2):181–186, June 2002.

[Lin *et al.*, 2007] Yi-Chien Lin, Li-Chun Sung, and Meng Chang Chen. An Automatic Multiple-Choice Question Generation Scheme for English Adjective Understandings. In *Workshop on Modeling, Management and Generation of Problems/Questions in eLearning, the 15th International Conference on Computers in Education (ICCE 2007)*, pages 137–142, 2007.

[Long, 1991] Michael H. Long. Focus on form: A design feature in language teaching methodology. *Foreign Language Research in Cross-cultural Perspective*, 1991.

[Long, 1996] Michael H. Long. The role of the linguistic environment in second language acquisition. *Handbook of Second Language Acquisition*, 1996.

[Macwhinney, 1995] Brian Macwhinney. Evaluating Foreign Language Tutoring Systems. *In Intelligent Language Tutors: Theory Shaping*, 50:317–326, 1995.

[Matthiessen and Bateman, 1991] Christian Matthiessen and John A. Bateman. *Text generation and systemic-functional linguistics: experiences from English and Japanese.* Pinter, 1991.

[McCoy *et al.*, 1992] Kathleen F. McCoy, K. Vijay-Shanker, and Gijoo Yang. A functional approach to generation with tag. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, ACL '92, pages 48–55, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[McKeown, 1992] Kathleen McKeown. *Text Generation.* Cambridge University Press, 1992.

[Melcuk, 1988] Igor A. Melcuk. *Dependency syntax : theory and practice.* SUNY series in linguistics. State University Press of New York, 1988.

[Meurers *et al.*, 2010] Detmar Meurers, Ramon Ziai, Luiz Amaral, Adriane Boyd, Aleksandar Dimitrov, Vanessa Metcalf, and Niels Ott. Enhancing authentic web

pages for language learners. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '10, pages 10–18, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Meurers, 2012] Detmar Meurers. Natural language processing and language learning. In Carol A. Chapelle, editor, *Encyclopedia of Applied Linguistics*. Blackwell, 2012. to appear.

[Michaud and McCoy, 2000] Lisa N. Michaud and Kathleen F. McCoy. Supporting intelligent tutoring in call by modeling the user's grammar. In *IN PROCEEDINGS OF THE 13TH ANNUAL INTERNATIONAL FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SYMPOSIUM*, pages 50–54. AAAI Press, 2000.

[Michaud *et al.*, 2000] Lisa N. Michaud, Kathleen F. McCoy, and Christopher A. Pennington. An intelligent tutoring system for deaf learners of written english. In *Proceedings of the fourth international ACM conference on Assistive technologies*, Assets '00, pages 92–100, New York, NY, USA, 2000. ACM.

[Mitchell *et al.*, 2011] Margaret Mitchell, Aaron Dunlop, and Brian Roark. Semi-supervised modeling for prenominal modifier ordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 236–241, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Mitkov *et al.*, 2006] Ruslan Mitkov, Le An Ha, and Nikiforos Karamanis. A computer-aided environment for generating multiple-choice test items. *Natural Language Engineering*, 12(2):177–194, 2006.

[Nagata, 1996] Noriko Nagata. Computer vs. workbook instruction in second language acquisition. *CALICO journal*, 14(1):53–75, 1996.

[Nakanishi *et al.*, 2005] Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic models for disambiguation of an hpsg-based chart generator. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 93–102, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[Nerbonne, 2003] John Nerbonne. *Handbook of Computational Linguistics*, chapter Natural language processing in computer-assisted language learning. Oxford University Press, 2003.

[Oepen and Carroll, 2000] Stephan Oepen and John Carroll. Ambiguity packing in constraint-based parsing: practical results. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 162–169, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[Ott *et al.*, 2010] Niels Ott, Detmar Meurers, and Universität Tübingen. Information retrieval for education: Making search engines language aware. themes in science and technology education. special issue on computer-aided language analysis, teaching and learning: Approaches, perspectives and applications 3(1–2), 9–30. Technical report, 2010.

[Pereira and Warren, 1983] Fernando C. N. Pereira and David H. D. Warren. Parsing as deduction. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, ACL '83, pages 137–144, Stroudsburg, PA, USA, 1983. Association for Computational Linguistics.

[Perez-Beltrachini *et al.*, 2012] Laura Perez-Beltrachini, Claire Gardent, and German Kruszewski. Generating Grammar Exercises. In *NAACL-HLT 7th Workshop on Innovative Use of NLP for Building Educational Applications*, Montreal, Canada, June 2012.

[Perez-Beltrachini, 2009] Laura Perez-Beltrachini. Using regular tree grammars to reduce the search space in surface realisation. Master's thesis, European Masters Program in Language and Communication Technologies, Nancy/Bolzano, 2009.

[Pienemann, 1998] Mamfred Pienemann. *Language processing and second language development: Processability theory*, volume 15. John Benjamins Publishing Company, 1998.

[Pietquin *et al.*, 2011] O. Pietquin, L. Daubigney, M. Geist, et al. Optimization of a tutoring system from a fixed set of data. In *Proceedings of the ISCA workshop on Speech and Language Technology in Education*, pages 1–4, 2011.

[Piwek and Boyer, 2012] P. Piwek and K. E. Boyer. Varieties of question generation. *Dialogue and Discourse, Special Issue on Question Generation*, 2012.

[Pogodalla, 2004] S. Pogodalla. Computing semantic representations: towards ACG abstract terms as derivation trees. In *Proceedings of TAG+7*, pages 64–71, 2004.

[Pollard and Sag, 1988] Carl Pollard and Ivan A. Sag. *Information-based syntax and semantics: Vol. 1: fundamentals*. Center for the Study of Language and Information, 1988.

[Power, 2011] Richard Power. Deriving rhetorical relationships from semantic content. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 82–90. Association for Computational Linguistics, 2011.

[Power, 2012] Richard Power. OWL Simplified English: A Finite-State Language for Ontology Editing. In Tobias Kuhn and Norbert E. Fuchs, editors, *Third International Workshop, CNL 2012*, Lecture Notes in Computer Science, pages 44–60, Zurich, Switzerland, August 2012. Springer-Verlag Berlin Heidelberg.

[Pravec, 2002] Norma A Pravec. Survey of learner corpora. *ICAME journal*, 26(81):114, 2002.

[Rambow and Joshi, 1994] Owen Rambow and Aravind Joshi. A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena. *Current issues in meaning-text theory*, 1994.

[Reiter and Dale, 1997] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3:57–88, 1997.

[Reiter, 1994] Ehud Reiter. Has a consensus nl generation architecture appeared, and is it psychologically plausible. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport ME, 1994.

[Rypa and Feuerman, 1995] Marikka Rypa and Ken Feuerman. Calle: An exploratory environment for foreign language learning. *Intelligent language tutors: theory shaping technology*, pages 55–76, 1995.

[Schabes *et al.*, 1988] Yves Schabes, Anne Abeille, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *IN PROCEEDINGS OF THE 12TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS (COLING'88)*, pages 578–583, 1988.

[Schmidt, 1995] Richard Schmidt. Consciousness and foreign language: A tutorial on the role of attention and awareness in learning. *Attention and awareness in foreign language learning*, 1995.

[Schmitz and Le Roux, 2008] S. Schmitz and J. Le Roux. Feature unification in TAG derivation trees. In C. Gardent and A. Sarkar, editors, *Proceedings of TAG+8*, pages 141–148, Tübingen, Germany, 2008.

[Scott *et al.*, 1998] Donia Scott, Richard Power, and Roger Evans. Generation as a solution to its own problem. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 256–265, 1998.

[Segond *et al.*, 2005] Frédérique Segond, Thibault Parmentier, Roberta Stock, Ran Rosner, and Mariola Usteran Muela. Situational language training for hotel receptionists. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pages 85–92, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[Shieber *et al.*, 1995] Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing. *JOURNAL OF LOGIC PROGRAMMING*, 1995.

[Shieber, 1985] S.M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343, 1985.

[Shieber, 1993] Stuart M. Shieber. The problem of logical-form equivalence. *Comput. Linguist.*, 19(1):179–190, March 1993.

[Shieber, 2006] S. Shieber. Unifying Synchronous Tree Adjoining Grammars and Tree Transducers via Bimorphisms. In *Proceedings of EACL*, pages 377–384, 2006.

[Siddharthan, 2010] A. Siddharthan. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 125–133, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Siddharthan, 2011] A. Siddharthan. Text simplification using typed dependencies: a comparison of the robustness of different generation strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 2–11, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[Simon *et al.*, 2010] Smith Simon, Avinesh P.V.S, and Kilgarriff Adam. Gap-fill Tests for Language Learners: Corpus-Driven Item Generation. In *Proceedings of ICON-2010: 8th International Conference on Natural Language Processing*, 2010.

[Smedt, 1990] Koenraad Jan Maria Johanna De Smedt. *INCREMENTAL SENTENCE GENERATION – A COMPUTER MODEL OF GRAMMATICAL ENCODING*. PhD thesis, 1990.

[Srinivas and Joshi, 1995] Bangalore Srinivas and Aravind K Joshi. Some novel applications of explanation-based learning to parsing lexicalized tree-adjoining grammars. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 268–275. Association for Computational Linguistics, 1995.

[Steedman, 2000a] Mark Steedman. Information structure and the syntax-phonology interface. *Linguistic inquiry*, 31(4):649–689, 2000.

[Steedman, 2000b] Mark Steedman. *The syntactic process*. MIT press, 2000.

[Sumita *et al.*, 2005] Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, EdAppsNLP 05, pages 61–68, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[Sykes *et al.*, 2008] Julie M. Sykes, Ana Oskoz, and Steven L. Thorne. Web 2.0, Synthetic Immersive Environments, and Mobile Resources for Language Education. *CALICO Journal*, 25(3):528–546, 2008.

[Vijay-Shanker and Joshi, 1988] K. Vijay-Shanker and AK Joshi. Feature Structures Based Tree Adjoining Grammars. *Proceedings of the 12th conference on Computational linguistics*, 55:v2, 1988.

[Vijay-Shanker *et al.*, 1987] K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. Characterizing Structural Descriptions Produced by Various Grammatical Formalisms. In *Proceedings of the 25th ACL*, pages 104–111, Stanford, California, USA, 1987.

[Vygotsky, 1986] Lev Vygotsky. *Thought and Language*. Cambridge, Mass.: The MIT Press, 1986.

[White, 2004] M. White. Reining in CCG chart realization. In *INLG*, pages 182–191, 2004.

[Whitelock, 1992] P. Whitelock. Shake-and-bake translation. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 784–791. Association for Computational Linguistics, 1992.

[Williams and Power, 2010] Sandra Williams and Richard Power. Grouping axioms for more coherent ontology descriptions. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 197–201. Association for Computational Linguistics, 2010.

[Winke and MacGregor, 2001] Paula Winke and David MacGregor. Review of Hot Potatoes, 2001.

[XTAG Research Group, 2001] XTAG Research Group. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, University of Pennsylvania, 2001.

[Yao *et al.*, 2012] X. Yao, G. Bouma, and Y. Zhang. Semantics-based question generation and implementation. *Dialogue and Discourse, Special Issue on Question Generation*, 2012.

[Zamorano-Mansilla, 2004] Juan Rafael Zamorano-Mansilla. Text generators, error analysis and feedback. In *InSTIL/ICALL 2004 Symposium on Computer Assisted Learning*, Venice, Italy, 2004.

[Zock and Quint, 2004] Michael Zock and Julien Quint. Converting an electronic dictionary into a drill tutor. In *InSTIL/ICALL 2004 Symposium on Computer Assisted Learning*, Venice, Italy, 2004.

[Zock, 1996] Michael Zock. Computational linguistics and its use in real world: the case of computer assisted-language learning. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 1002–1004, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.