Analyse syntaxique profonde

4.1 Introduction

Cette dernière partie présente quelques unes de nos expérimentations concernant *l'analyse syntaxique profonde* de textes en exploitant les propriétés syntaxiques des éléments prédicatifs du français encodées dans les tables du lexique-grammaire. A cet effet, nous avons développé une grammaire lexicalisé du français dans un formalisme à unification original (*RTN décorés*), qui est générée de manière semi-automatique à partir des tables du lexique-grammaire.

Nous présentons, dans un premier temps, la méthodologie du lexique-grammaire, en précisant en quoi les tables du lexique-grammaire sont une ressource précieuse pour le traitement automatique des langues. Nous présentons ensuite le formalisme grammatical des RTN décorés et nous le situons par rapport à d'autres formalismes utilisés couramment pour l'analyse syntaxique. Puis, nous présentons notre grammaire du français dans son état actuel et nous terminons en donnant quelques résultats préliminaires des évaluations de sa couverture lexicale et syntaxique.

4.2 Lexique-grammaire

Le lexique grammaire est une méthodologie pour l'étude empirique de la syntaxe des langues naturelles créée par Maurice Gross dont l'ouvrage fondateur est *Méthode en syntaxe* [Gross, 1975]. Cette méthodologie a pour cadre théorique la grammaire transformationnelle Harissienne [Harris, 1951, Harris, 1968] qui consiste en une approche mathématique de la linguistique, reposant sur des définitions rigoureuses et minimales. Dans ce cadre, le sujet d'étude est la phrase simple qui est considérée comme l'unité minimale de sens. Une phrase simple est composée d'un prédicat et de ses actants ou arguments, c'est-à-dire son sujet et ses compléments essentiels (par opposition aux compléments dits non-essentiels ou circonstanciels). Le prédicat est le noyau de la phrase, il s'agit le plus souvent d'un verbe plein. Ses arguments se distinguent par leur position dans la phrase (sujet, complément d'objet direct, complément prépositionnel, etc.) et par leur nature (groupe nominal humain, concret, phrase complétive, infinitive, etc.) :

Luc mange une pomme
Luc donne une pomme à Lea
(Que Lea soit partie+Partir en vacances) rejouit Max

Chaque prédicat sélectionne le nombre et la nature de ses arguments comme en attestent les phrases suivantes, qui sont grammaticalement incorrectes :

**Luc mange une pomme à Lea*
**Que Lea soit partie donne Max*
**Luc rejouit une pomme*

Le prédicat d'une phrase élémentaire peut être également un nom, un adjectif ou un adverbe. Dans ce cas, la phrase contient également un verbe considéré comme sémantiquement vide et appelé *verbe support* dans ce contexte :

Luc prend la *douche* (= la douche que Luc prend)
Lea est *fière* de son fils
Cet évenement s'est déroulé *(tranquillement+jeudi dernier)*

Certaines expressions dites figées peuvent également jouer le rôle de prédicat d'une phrase. Elles ne présentent en surface pas de différences avec les constructions dites libres, mais certains de leurs éléments sont contraints et n'admettent pas ou peu de variations lexicales :

Luc prend la tangente (= * la tangente que Luc prend)
Luc prend le taureau par les cornes
Lea fait face à ce problème

Une phrase simple peut être sujette à une transformation (telle que la mise au passif, ou la pronominalisation, l'effacement ou le deplacement d'un argument par exemple) produisant ainsi une nouvelle phrase considérée comme transformationnellement équivalente :

Luc a rendu la télé à Lea
= *Luc a rendu à Lea la télé [permutation]*
= *La télé a été rendue à Lea par Luc [passif]*
= *La télé a été rendue à Lea [effacement]*
= *La télé lui a été rendue [pronominalisation]*

D'autre type de transformations, dites transformations binaires, permettent de construire des phrases complexes à partir de deux phrases simples (ou complexes). Il s'agit par exemple

– de la coordination :

Luc danse ; Marie s'ennuie
= *Luc danse et Marie s'ennuie*

– de la subordination :

Luc danse tandis que Marie s'ennuie

– de la relativation :

Luc aime la danse ; Lea exècre Luc
= *Luc, que Lea exècre, aime la danse*

Selon Harris, les phrases simples sont des unités élémentaires qui permettent de construire tous types d'énoncés complexes par applications successives

de transformations. Inversement, tout énoncé en langue naturelle peut être décomposé en un ensemble de phrases élémentaires qui sont les unités de sens qui le composent.

Maurice Gross a mis en avant l'importance du lexique par rapport à la grammaire. En effet, comme même Chomsky l'observait [Chomsky, 1965], les transformations syntaxiques, même les plus générales, sont sujettes à de fortes contraintes au niveau lexical. Ainsi, une description complète de la syntaxe d'une langue naturelle ne consiste pas en un ensemble de règles syntaxiques générales mais nécessite également, et de manière aussi importante, une description détaillée pour chaque élément du lexique des formes et des transformations syntaxiques qu'il accepte ou n'accepte pas. Ainsi, Maurice Gross a entrepris, avec son équipe au LADL, une description des propriétés syntaxiques des éléments prédicatifs du français : verbes, noms, adjectifs, adverbes et phrases figées. Pour chaque prédicat, ont été étudiées de manière sytématique ses propriétés de sous-catégorisation (le nombre et la nature de ses arguments) ainsi que ses propriétés transformationnelles. Toutes ces descriptions ont été encodées dans des dictionnaires syntaxiques sous la forme de tables, dites tables du lexique grammaire. Chaque table regroupe un ensemble de prédicats en fonction de propriétés définitionnelles, correspondant souvent à la structure de la phrase canonique. La figure 4.1, par exemple, contient un extrait de la table 9 qui regroupe l'ensemble des verbes qui rentrent dans la construction *N0 V que P à N1 (= : Luc dit qu'il va bien à Lea)*. Chaque ligne de la table correspond à un prédicat ; chaque colonne correspond à une propriété. Une valeur booléenne (indiquée par un + ou −) à chaque intersection d'une ligne et d'une colonne indique si telle entrée accepte ou non telle propriété. Les chercheurs du lexique-grammaire ont ainsi codé 12 000 emplois de verbes [Gross, 1975, Boons *et al.*, 1976b, Boons *et al.*, 1976a, Guillet et Leclère, 1992], 10 000 emplois de noms prédicatifs [Giry-Schneider, 1978][Labelle, 1974][Meunier, 1981][Vivès, 1983] [de Negroni-Peyre, 1978 [Gross, 1989]. Les tables des adjectifs sont en cours de construction et il existe également des tables de phrases figées (M. Gross, 1984) comprenant une vingtaine de milliers d'entrées.

Le but de notre travail consiste à exploiter les informations de sous-catégorisation et les propriétés transformationnelles présentes dans les tables du lexique-grammaire afin d'identifier les phrases simples (c'est-à-dire les prédicats syntaxiques et leurs arguments) dans les textes français. Par exemple,

N0 = : Nhum	N0 = : Nnr	N0 = : le fait que P	N0 = : V1 W	N0 = : V2 W	Aux = : avoir	entry	N0 V	N0 V W	N0 pousser Dét V-m	N0 pousser Dét V-E	N0 V contre Nhum	N0 V après Nhum	N0 V N1	N1 = : que P	N1 = : V0 W	N1 = : Aux V0 W	N1 = : de V0 W	(N1) (être Adj)	(être Adj) (que P)	que N0 V (être Adj)	(N1) (Adj)	(Adj) (que P)	que N0 V (Adj)	Nég interr = > subj	Impératif = > subj	N1 = : V0 W	N1 = : V2 W
+	-	-	-	-	+	développer	-	-	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	+	-	-	-
+	-	-	-	-	+	dévider	-	-	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	+	-	-	-
+	+	+	+	+	+	dévoiler	-	-	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	-	-	-	-
+	+	+	-	+	+	dicter	-	+	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	+	-	+	-
+	-	-	-	-	+	dicter	-	-	-	-	-	-	+	+	-	+	-	-	-	-	-	-	-	-	-	+	-
+	+	+	-	+	+	diffracter	-	-	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	+	-	+	-
+	-	-	-	-	+	diffuser	-	-	-	-	-	-	+	+	-	+	-	-	-	-	-	-	-	-	-	-	-
+	+	+	-	+	+	dire	-	-	-	-	-	-	+	+	+	+	-	-	+	+	-	-	-	+	-	+	-
+	+	+	-	+	+	disputer	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-
+	+	+	+	+	+	dissimuler	-	-	-	-	-	-	+	+	-	+	-	-	-	-	-	-	-	+	-	-	-
+	+	+	-	+	+	divulguer	-	-	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	-	-	-	-
+	-	-	-	-	+	donner	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
+	+	+	-	+	+	économiser	-	+	-	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-	+	-
+	-	-	-	-	+	écrire	-	+	-	-	+	-	-	+	-	+	-	-	+	+	-	-	-	+	-	+	-
+	-	-	-	-	+	édicter	-	-	-	-	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	+	-
+	-	-	-	-	+	émettre	-	-	-	-	-	-	+	+	-	+	-	-	+	+	+	-	+	+	-	-	-
+	+	+	-	+	+	enjoindre	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-
+	+	+	-	+	+	enlever	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	-
+	-	-	-	-	+	énoncer	-	-	-	-	-	-	+	+	-	+	-	-	+	+	-	-	-	+	-	+	-

FIG. 4.1 – Extrait de la table 9

étant donné le texte suivant (extrait de [Goscinny et Sempé, 2006]) :

> Ce matin, Geoffroy est entré dans la cour de l'école, il s'est arrêté et il nous a crié : «Eh ! Les Gars ! Venez voir ce que j'ai !» C'est drôle, chaque fois que Geoffroy amène quelque chose à l'école, il s'arrête à l'entrée de la cour de récré, il nous crie : « Eh ! Les Gars ! Venez voir ce que j'ai ! » Alors, nous y sommes allés et il avait un stylo.

Nous souhaitons en extraire automatiquement les phrases élementaires qui le composent (afin de simplifier notre exemple, nous considérons la séquence de phrases au discours direct comme une seule unité, et nous omettons de

reporter les temps de conjugaison) :

P' : *«Eh ! Les Gars ! Venez voir ce que j'ai ! »*
P1 : *Geoffroy entre dans la cour de l'école*
P2 : *Geoffroy s'arrête*
P3 : *Geoffroy nous crie P'*
P4 : *P1, P2 et P3 se passe ce matin*
P5 : *Geoffroy amène quelque chose à l'école*
P6 : *P1, P2 et P3 se passe chaque fois que P5*
P7 : *P6 est drôle*
P8 : *nous allons à l'entrée de la cour de récré*
P9 : *Geoffroy a un stylo*

Nous pouvons également représenter ces phrases simples sous la forme de prédicats logiques :

P1 : `entrer(Geoffroy, dans(cours de l'école))`
P2 : `se-arrêter(Geoffroy)`
P3 : `crier(Geoffroy, P')`
P4 : `ce-matin(et(P1,P2,P3))`
P5 : `amener(Geoffroy, quelque chose, école)`
P6 : `chaque-fois(et(P1, P2, P3), P5)`
P7 : `drôle(P6)`
P8 : `aller(nous, entrée de la cour de récré)`
P9 : `avoir(Geoffroy, stylo)`

Nous n'avons pas résolu tous les problèmes nous permettant de réaliser une telle analyse automatiquement, cependant, les travaux présentés dans ce chapitre nous ont permis d'avancer sérieusement dans ce sens.

L'analyse syntaxique ne constitue pas une fin en soi, mais est souvent une étape nécessaire pour le développement d'applications de plus haut niveau en traitement des langues, tels que l'analyse sémantique, les systèmes de réponses à des questions, la génération de résumés de texte ou la traduction automatique. Toutes ces applications pourraient bénéficier d'un module d'analyse syntaxique exploitant les descriptions linguistiques fines et à large

couverture telles que celles qui ont été élaborées dans le cadre du lexique-grammaire [Gardent *et al.*, 2005].

4.3 Formalisme Grammatical

4.3.1 Stuctures de traits et unification

Une structure de traits est constituée d'un ensemble de traits sous la forme de couples attribut-valeur. La valeur d'un trait peut être de deux formes. Il s'agit soit d'une valeur atomique (dans ce cas elle prend la forme d'un symbole ou plus généralement d'une disjonction de symboles atomiques), soit d'une structure de traits enchâssée. Traditionnellement, on représente une structure de traits par des matrices entre crochets. La figure 4.2 présente deux exemples de structures de traits. La première est une structure de traits simple dans le sens où chaque trait à une valeur atomique. Le trait *cat* par exemple a pour valeur le symbole *det* et la valeur de l'attribut *genre* est constituée de la disjonction des deux symboles atomiques *masc* et *fem*. La matrice à droite, quant à elle, représente une structure de traits complexe puisque la valeur du trait *accord* est elle-même une structure de traits enchassée.

$$\begin{bmatrix} cat : det \\ genre : masc|fem \\ nombre : pl \end{bmatrix} \quad \begin{bmatrix} cat : det \\ accord : \begin{bmatrix} genre : masc \\ nombre : plur \end{bmatrix} \end{bmatrix}$$

FIG. 4.2 – Exemples de structures de traits

Pour faciliter la formalisation de certains phénomènes de syntaxe, nous manipulerons des structures de traits dont la valeur d'un trait peut être également un ensemble de valeurs (atomiques ou complexes). Cette extension ne modifie pas notre définition des structures de traits puisque nous représentons ce type de valeur par des structures de traits enchassées à deux attributs *head* et *tail*, à la manière d'une liste chaînée (cf. figure 4.3). Pour des raisons de lisibilité, nous affichons ce type de valeur sous la forme d'une liste encadrée par les symbole $<$ et $>$:

$$\begin{bmatrix} head : Pierre \\ tail : \begin{bmatrix} head : Paul \\ tail : \begin{bmatrix} head : Jacques \\ tail : [] \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad \text{<Pierre,Paul,Jacques>}$$

FIG. 4.3 – Représentations des traits à valeur ensembliste

L'ordre dans lequel apparaissent les attributs dans une structure de trait n'est pas important. Par exemple, les deux structures de traits suivantes sont strictement équivalentes.

$$\begin{bmatrix} cat : det \\ accord : \begin{bmatrix} genre : masc \\ nombre : plur \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} accord : \begin{bmatrix} nombre : plur \\ genre : masc \end{bmatrix} \\ cat : det \end{bmatrix}$$

FIG. 4.4 – Structure de traits équivalentes

Une structure de traits ne peut pas avoir deux traits distincts pour un même attribut. Par exemple, la matrice à gauche dans la figure 4.5 ne représente pas une structure de traits bien formée ; la structure de droite est bien formée puisque les deux attributs *genre* n'apparaissent pas au même niveau d'enchâssement dans la structure.

$$\begin{bmatrix} genre : masc \\ genre : fem \end{bmatrix} \quad \begin{bmatrix} genre : masc \\ accord : [\ genre : fem\ ] \end{bmatrix}$$

FIG. 4.5 – Exemple de structures de traits mal-formées et bien formées

Enfin, plusieurs traits peuvent partager une même valeur dans une structure de traits. Dans ce cas la structure de traits est dite *réentrante*. Par exemple dans la structure de trait A de la figure 4.6, les deux attributs *accord* sont coïndicés par l'indice 1 et partagent donc le même trait $[\ num : sing\ ]$.

$$A$$
$$\left[\begin{array}{l} Det : \left[\ accord : (1) \left[\ num : sing\ \right]\ \right] \\ N : \left[\ accord : (1)\ \right] \end{array}\right]$$

FIG. 4.6 – Structure de traits A réentrante

Cette structure est différente de la structure B, dans laquelle les valeurs de ses deux attributs *accord* sont identiques mais ne sont pas partagées.

$$B$$
$$\left[\begin{array}{l} Det : \left[\ accord : \left[\ num : sing\ \right]\ \right] \\ N : \left[\ accord : \left[\ num : sing\ \right]\ \right] \end{array}\right]$$

FIG. 4.7 – Structure de trait B non réentrante

Subsomption

La relation de *subsomption* définit une relation d'ordre partiel sur l'ensemble des structures de traits.

On dit qu'une structure de traits A subsume une structure de traits B (ce qui est noté $A \sqsubseteq B$) si et seulement si :

- tous les traits à valeur atomique présents dans A sont présents dans B avec la même valeur ou une valeur plus précise.
- pour tout trait à valeur non atomique dans A, ce trait est présent dans B et la valeur de ce trait dans A subsume la valeur de ce même trait dans B ;
- si deux traits ont une valeur partagée dans A, alors ces mêmes traits ont une valeur partagée dans B.

Informellement, on dit que A subsume B si toutes les informations contenues dans A sont également contenues dans B ou encore si B est plus (ou également) précise que A.

Par exemple, étant données les structures de traits suivantes (figure 4.8) :

$$\underset{F_1}{\left[\ genre : masc|fem\ \right]} \qquad \underset{F_2}{\left[\begin{array}{l} num : pl \\ genre : masc|fem \end{array}\right]} \qquad \underset{F_3}{\left[\begin{array}{l} num : pl \\ genre : masc \end{array}\right]}$$

FIG. 4.8 – Exemple de subsomptions

La structure F_1 subsume F_2 et F_3 mais F_2 et F_3 ne subsument pas F_1 puisqu'elles contiennent un attribut *genre* qui est absent dans F_1. De même F_2 subsume F_3 mais F_3 ne subsume F_2 puisque la valeur de l'attribut *genre* est plus précise dans F_3 que dans F_2.

La relation de subsomption définit une relation d'ordre partielle sur l'ensemble des structures de traits, dans le sens où elle ne permet pas de mettre en relation tous couples de structures de traits. En effet, étant données les structures de traits suivantes :

$$\left[\begin{array}{l} num : sg \\ genre : masc \end{array}\right] \qquad \left[\begin{array}{l} person : 3 \\ num : sg \end{array}\right]$$

FIG. 4.9 – Structures de traits qui ne se subsument pas

aucune des structures ne subsume l'autre, puisque chacune contient une information absente dans l'autre structure.

Enfin, étant données les structure A et B vues précédemment (figures 4.6 et 4.7), la structure B subsume la structure A mais A ne subsume pas B. En effet, toutes les informations présentes dans B sont bien présentes dans A, en revanche, les deux attributs *accord* enchassés dans les attributs DET et N de A partagent la même valeur, ce qui n'est pas le cas dans la structure B.

Unification

L'unification est une opération fondamentale sur les structures de traits ; elle permet de combiner les informations présentes dans deux structures de traits lorsqu'elles sont compatibles.

Formellement, étant données deux structures de traits A et B, le résultat de l'unification de A et B (notée $A \sqcup B$) est la structure de traits minimale F telle que A et B subsument F. Si une telle structure n'existe pas, l'unification échoue et son résultat est notée $\bot$.

Informellement, l'unification de deux structures de traits produit la plus petite structure de traits qui contient toutes les informations contenues dans les deux structures, si ces informations sont compatibles.

Par exemple, étant donné les structures F_1 et F_2 suivantes

$$F_1 \qquad\qquad F_2$$
$$\begin{bmatrix} cat : N \\ accord : \begin{bmatrix} num : sg \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} cat : N \\ accord : \begin{bmatrix} genre : fem \end{bmatrix} \end{bmatrix}$$

FIG. 4.10 – Structures F_1 et F_2

le résultat de l'unification de ces deux structures est la structure suivante :

$$F_1 \sqcup F_2$$
$$\begin{bmatrix} cat : N \\ accord : \begin{bmatrix} num : sg \\ genre : fem \end{bmatrix} \end{bmatrix}$$

FIG. 4.11 – $F_1 \sqcup F_2$

En revanche, étant données les deux structures suivantes :

$$\begin{array}{cc} F_3 & F_4 \\ \left[\ cat : N\ \right] & \left[\ cat : Det\ \right] \end{array}$$

FIG. 4.12 – Structure de traits incompatibles

l'unification de ces deux structures échoue, car il n'existe aucune structure de traits qui contienne les informations contenues dans F_3 et F_4, car celles-ci sont contradictoires. $F_3 \sqcup F_4 = \bot$.

Enfin, si nous reprenons les structures A réentrante et B non réentrante des figures 4.6 et 4.7 et que l'on unifie ces deux structures avec la structure C ci-dessous, les résultats seront bien différents :

$$C$$
$$\left[\ Det : \left[\ accord : \left[\ genre : masc\ \right]\ \right]\ \right]$$
$$A \sqcup C :$$
$$\left[\begin{array}{l} Det : \left[\ accord : 1 \left[\begin{array}{l} num : sing \\ genre : masc \end{array}\right]\ \right] \\ N : \left[\ accord : (1)\ \right] \end{array}\right]$$
$$B \sqcup C :$$
$$\left[\begin{array}{l} Det : \left[\ accord : \left[\begin{array}{l} num : sing \\ genre : masc \end{array}\right]\ \right] \\ N : \left[\ accord : \left[\ num : sing\ \right]\ \right] \end{array}\right]$$

FIG. 4.13 – Résultats de l'unification

Ainsi, dans la structure obtenue par unification de A et B, le trait *genre* enchassé sous l'attribut *accord* de l'attribut N est spécifié (et partage sa valeur avec le même attribut enchassé sous l'attribut *Det*), ce qui n'est pas le cas pour la structure obtenue après unification de B et C.

4.4 Les RTN décorés comme formalisme grammatical

Le formalisme des RTN décorés (DRTN) est une extension du formalisme WRTN dans lequel nous avons intégré une composante d'unification. Une grammaire DRTN est une grammaire WRTN dont les sorties sont décorées par des équations sur les traits. Ces équations permettent de construire les structures de traits associées à chaque constituant syntaxique lors de l'analyse. Comme nous le verrons, l'utilisation des structures de traits nous a permis de formaliser et résoudre différents phénomènes de syntaxe, des plus simples comme les contraintes d'accord grammatical jusqu'à des phénomènes plus complexes tels que la résolution de certaines coréférences ou les dépendances à distance.

Le résultat de l'analyse d'une phrase par une grammaire DRTN est constitué d'une forêt d'arbres syntaxiques qui représentent les possibles découpages de la phrase en constituants syntaxiques ; à chaque arbre est associée une structure de traits dans laquelle sont stockées les relations grammaticales entre ces constituants qui ont été calculées durant l'analyse à l'aide des équations sur les traits qui décorent les transitions de la grammaire. La représentation du résultat de l'analyse par une structure de traits nous permet plus d'expressivité qu'avec un simple arbre syntaxique. En particulier, nous nous en servons, dans le cadre de ce travail, pour identifier dans les textes les actants sémantiques de chaque prédicat indépendamment de leur position syntaxique dans la phrase.

Par exemple, la figure 4.17 et les figures suivantes (page 127) présentent une grammaire simplifiée pour le verbe *donner*. Le format et la sémantique des équations sur les traits présentes dans les sorties de la grammaire seront expliqués plus en détail dans la suite.

Cette grammaire analyse par exemple, les deux phrases suivantes :

Luc a escroqué une forte somme à Lea
Luc a escroqué Lea d'une forte somme

Ces deux constructions sont considérées, dans le cadre du lexique-grammaire, comme transformationnellement équivalentes ; la seconde étant obtenue comme le résultat de l'application d'une transformation *de croisement des arguments* sur la première. Ainsi, les actants *Luc*, *Lea* et *somme* jouent le même rôle argumental par rapport au prédicat *escroquer* dans les deux phrases ; cette équivalence n'apparaît pas au niveau des arbres syntaxiques obtenus à l'issue de l'analyse de ces deux phrases (figures 4.14 et 4.15) :

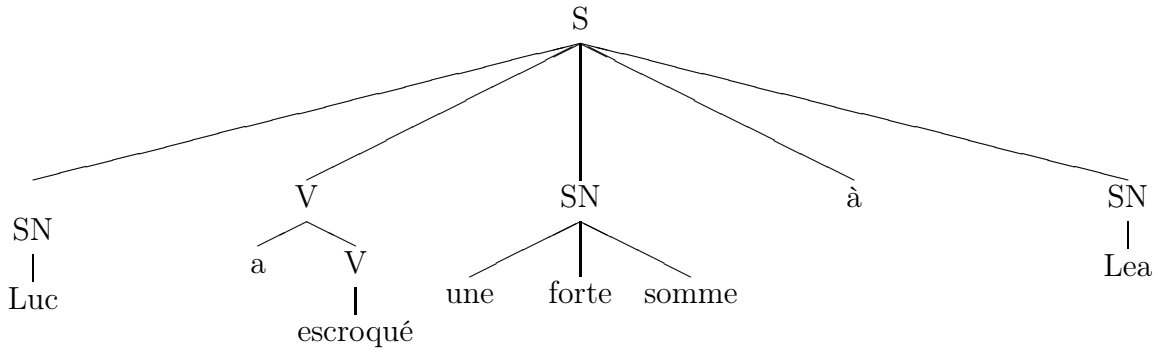

FIG. 4.14 – Arbre syntaxique associé à la phrase *Luc a escroqué une forte somme à Lea*

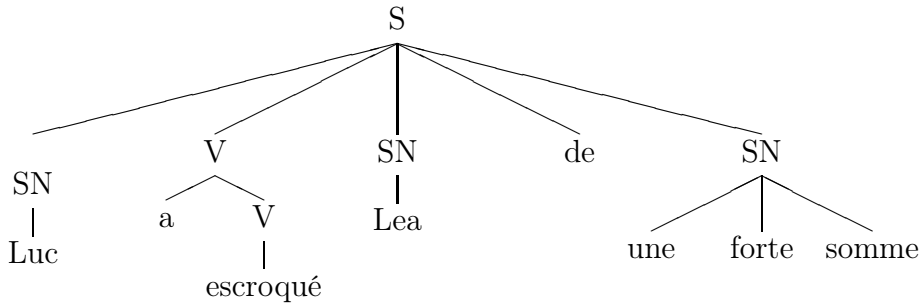

FIG. 4.15 – Arbre syntaxique associé à la phrase *Luc a escroqué Lea d'une forte somme*

En revanche, à l'issue de l'analyse, nous associons à chacun de ces arbres de dérivation une structure de traits dans laquelle ces relations d'équivalence

entre le prédicat et ses arguments sont bien explicitées, puisque nous obtenons exactement la même structure de traits comme résultat de l'analyse des deux phrases :

$$\begin{bmatrix} Pred : (1)escroquer \\ V : \begin{bmatrix} lemma : (1) \\ mode : ind \end{bmatrix} \\ n0 : Luc \\ n1 : somme \\ n2 : Lea \end{bmatrix}$$

FIG. 4.16 – Structure de traits obtenue comme résultat de l'analyse des deux phrases transformationnellement équivalentes

C'est en ce sens que nous parlons de syntaxe *profonde*.

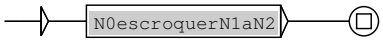

FIG. 4.17 – Graphe non terminal P (axiome de la grammaire)

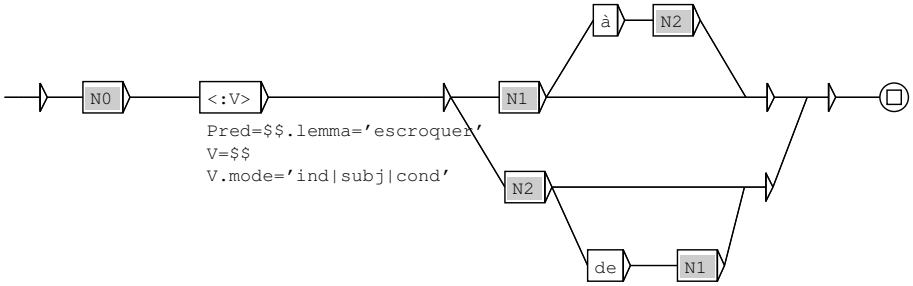

FIG. 4.18 – Sous-graphe de structuration N0escroquerN1aN2

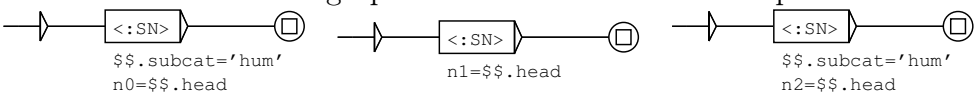

FIG. 4.19 – Sous-graphes N0, N1, N2

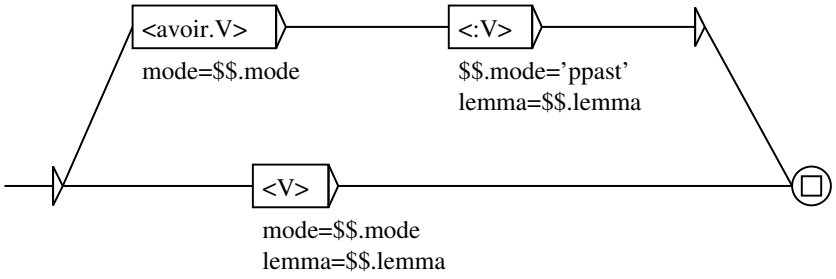

FIG. 4.20 – Non terminal V

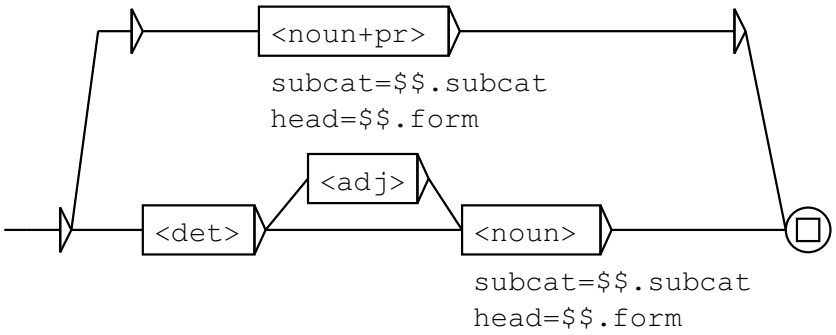

FIG. 4.21 – Non terminal SN

Les différentes conventions d'appel à un sous-graphe

Nous faisons la distinction dans notre formalisme entre *sous-graphe de structuration* et *sous-graphe non terminal*. Ces deux types de graphes se distinguent par la forme avec laquelle ils sont instanciés depuis leur graphe parent (voir par exemple notre grammaire jouet page 127) :

- la convention classique d'appel à un sous-graphe (notée :X) qui apparait en grisé dans les figures ; ce type d'appel à un sous-graphe ne correspond pas à la reconnaissance d'un non-terminal de la grammaire. En effet, lors de la compilation de celle-ci, les sous-graphes appelés avec ce type de convention sont directement intégrés dans leur graphe parent à l'aide des méthodes d'aplatissement de grammaire que nous avons décrites dans la section 3.4.
- l'appel à un non-terminal noté < :X> ; dans ce cas X est un symbole non terminal dont les formes sont décrites dans un automate de la grammaire.

Les sous-graphes non terminaux servent aux appels (éventuellement récursifs) du RTN et correspondent à des noeuds de l'arbre de dérivation. Les sous-graphes de structuration servent à assurer la compacité et la lisibilité des éléments de la grammaire et ne peuvent pas s'appeler récursivement entre eux (sauf par l'intermédiaire d'un sous-graphe non terminal).

On peut voir que le graphe `N0escroquerN1aN2` (figure 4.18) utilise les deux types de conventions d'appel : les 3 arguments du verbe sont reconnus dans des sous-graphes appelés par la convention classique :N0, :N1, :N2 ; alors que le noyau verbal de la phrase est reconnu par le constituant syntaxique V de la grammaire décrit dans la figure 4.20.

La figure 4.22 présente l'arbre syntaxique obtenu après l'analyse de la phrase : *Cet homme a escroqué une fleur à Lea.*

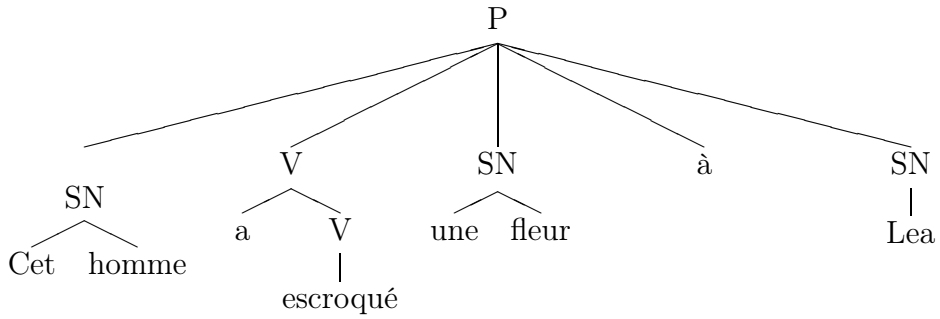

FIG. 4.22 – Arbre de dérivation

On constate que les non terminaux de la grammaire apparaissent comme les noeuds internes de l'arbre de dérivation, ce qui n'est pas le cas des sous-graphes `N0escroquerN1aN2`, `N0`, `N1` et `N2` qui eux n'apparaissent pas.

Par ailleurs, si l'on souhaite enrichir notre grammaire de nouvelles constructions, les sous-graphes décrivant les actants du verbe peuvent directement être réutilisés, comme par exemple dans le graphe escroquer-passif de la figure 4.23 qui décrit certaines constructions à la voix passive pour le verbe *escroquer*.

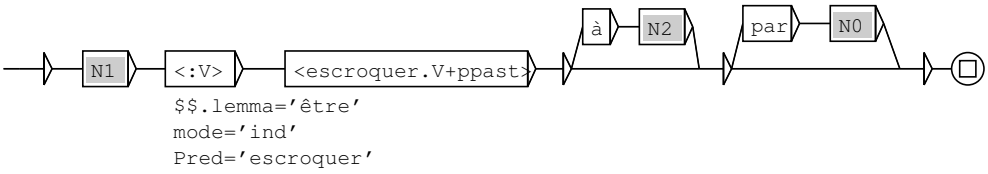

FIG. 4.23 – Constructions passives

Nous modifions également le graphe principal P, axiome de la grammaire, de manière à ce qu'il prenne en compte ces nouvelles descriptions (cf. figure 4.24). Notre grammaire analyse maintenant de nouvelles constructions telles que :

> *une (belle+sacrée) somme a été escroquée à l'état (par le premier ministre+E).*

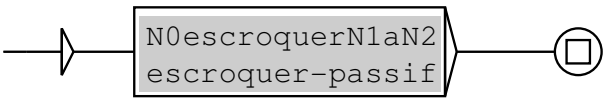

FIG. 4.24 – Graphe principal modifié

Ainsi, grâce à ce système à deux conventions d'appel à des sous-graphes, nous obtenons une indépendance entre la structuration de la grammaire en graphes et la structure des arbres syntaxiques. Ceci permet à l'auteur de la grammaire de factoriser certaines descriptions linguistiques dans des graphes en vue de leur réutilisation dans des descriptions de tailles plus importantes sans que ces choix influencent directement la forme des structures produites lors de l'analyse.

Equations sur les traits

Les équations sur les traits (ou *équations fonctionnelles*) spécifient des contraintes sur les structures de traits qui sont associées à chaque constituant syntaxique reconnu par la grammaire. Ces contraintes permettent de construire ces structures de traits durant l'analyse ou éventuellement de faire échouer l'analyse dans le cas où des contraintes ne peuvent pas être satisfaites.

Les équations utilisent la notion de *chemin* dans une structure de traits (ou *chemin fonctionnel*) pour identifier un trait dans une structure de traits. Un chemin fonctionnel consiste en la séquence des attributs (séparés par un '.') qu'il faut franchir en partant de la racine de la structure de traits pour parvenir à un trait particulier. Par exemple, étant donnée la structure suivante :

$$\left[\begin{array}{l} det : (1)\left[\ accord : \left[\begin{array}{l} genre : masc \\ num : pl \end{array}\right]\ \right] \\ n : (1) \end{array}\right]$$

Le chemin *det.accord.genre* identifie la valeur du trait d'attribut *num* enchassé dans l'attribut *accord* enchassé dans l'attribut *det*. Ce même trait est d'ailleurs identifié par le chemin *n.accord.genre* puisque la structure est réentrante et que ces deux traits partagent la même valeur.

Par défaut, un chemin dans une équation fonctionnelle référence un trait dans la structure de traits associée au non terminal décrit par le graphe courant. Cependant, le symbole $$ permet de référencer la structure de trait associée au symbole contenu dans la boîte sous laquelle l'équation apparaît. Par exemple dans le graphe de reconnaissances des SN de la figure 4.21, l'équation *subcat=$$.subcat* sous la boîte étiquetée par <noun> permet d'unifier la valeur de l'attribut *subcat* de la structure de traits associée au non terminal courant SN (chemin : *subcat*) avec la valeur de l'attribut *subcat* de la structure associée au nom reconnu par le masque lexical <noun> (chemin : *$$.subcat*).

Dans le cas où le symbole qui étiquette la transition est un symbole non terminal (de la forme < :X>), la structure de traits référencée par le symbole $$ est la structure associée au non terminal X reconnu lors du franchissement de la transition ; cette structure est construite durant l'analyse de ce constituant.

Dans le cas où le symbole qui étiquette la transition est un symbole terminal (c'est-à-dire un masque lexical), le symbole $$ référence la structure de traits qui est construite à partir du mot étiqueté du texte (représenté par un masque lexical) qui a permis le franchissement de cette transition. Un masque lexical étant lui-même représenté par une structure composée de champs sous la forme de couples attribut-valeur, la transformation d'un masque en une structure de traits est triviale : l'opération consiste à reformater ces informations sous la forme d'une structure de traits simple dont les traits ont une valeur atomique, les noms des attributs étant décrits dans la description du jeu d'étiquettes. Par exemple, le mot étiqueté <corps,corps.noun+conc+m+s+p> se traduit sous la forme de la structure suivante :

$$\begin{bmatrix} form : corps \\ lemma : corps \\ CAT : noun \\ subcat : conc \\ gender : m \\ number : s|p \end{bmatrix}$$

FIG. 4.25 – Structure de traits associée au nom *corps*

L'utilisation du symbole $$ dans une équation fonctionnelle n'a pas de sens si elle est présente dans les sorties d'une boîte qui contient un appel à un sous-graphe de structuration (de la forme :X). En effet, comme on l'a vu précédemment, un graphe appelé par cette convention est directement intégré dans son graphe parent et par conséquent ne décrit pas un constituant syntaxique de la grammaire. A ce titre, il n'a pas de structure associée propre et les équations fonctionnelles qu'il contient contribuent directement à la construction de la structure de traits associée au constituant syntaxique décrit dans le graphe à partir duquel il a été intégré.

Les équations sur les traits qui décorent les grammaires DRTN peuvent être de plusieurs formes :

contrainte d'unification

Les contraintes d'unification sont la forme la plus courante des équations fonctionnelles. C'est d'ailleurs la seule forme que nous avons utilisée dans nos exemples jusqu'à présent. Une contrainte d'unification a une des deux formes suivantes :

– $chemin_1 = chemin_2$
– $chemin_1 =$ *'valeur'*

où $chemin_1$ et $chemin_2$ sont des chemins fonctionnels et *valeur* est un symbole atomique ou une disjonction de symboles atomiques. Une contrainte de ce type permet d'unifier la valeur de deux traits spécifiés par des chemins

fonctionnels (première forme), ou d'unifier la valeur d'un trait avec une valeur atomique précisée dans la grammaire (seconde forme). Dans le cas où l'unification échoue parce que les valeurs sont incompatibles, l'analyse échoue.

Par exemple, l'équation *$$.subcat='hum'* sous la boîte étiquetée < :SN> dans le graphe N0 (figure 4.19) permet de contraindre la reconnaissance aux constituants de catégorie SN ayant un attribut *subcat* compatible avec la valeur *hum* ; cet attribut étant lui-même hérité du mot de tête du SN par l'équation *subcat=$$.subcat* (figure 4.21).

De même, nous pouvons améliorer le graphe SN de manière à vérifier l'accord en genre et en nombre entre le déterminant, l'éventuel adjectif et le nom de tête et faire remonter ces informations au niveau de la structure de traits associée à ce constituant. Le graphe devient alors celui de la figure 4.26.

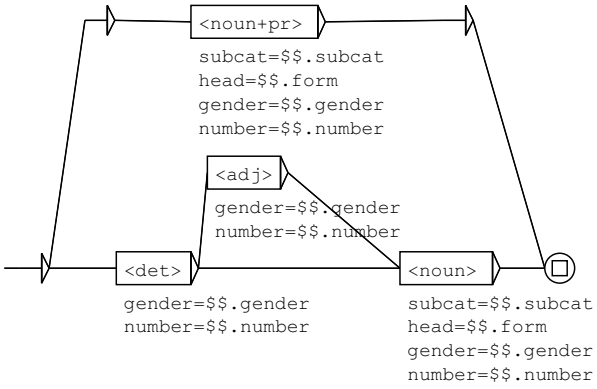

FIG. 4.26 – Vérification de l'accord grammatical au sein d'un SN

Ainsi, ce graphe SN reconnaîtra par exemple la séquence étiquetée

<les,le.det+def+p> <vauriens,vaurien.noun+hum+m+p>

et associera à son analyse la structure de traits suivante :

$$\begin{bmatrix} CAT : SN \\ head : vauriens \\ subcat : hum \\ gender : m \\ number : p \end{bmatrix}$$

En revanche, ce même graphe n'analysera pas la séquence :

<les,le.det+def+p> <vaurien,vaurien.noun+hum+m+s>

puisque les valeurs *p* et *s* pour le trait *number* sont incompatibles et ne peuvent pas s'unifier.

contraintes existentielles

Les contraintes existentielles sont une autre forme d'équation sur les traits qui peuvent décorer une grammaire DRTN. Ce type d'équation permet d'imposer la présence (ou l'absence) d'un attribut sans spécifier sa valeur.

Une contrainte existentielle a une des deux formes suivantes :

– *chemin*
– *˜chemin*

La première forme, constituée d'un chemin fonctionnel, spécifie que l'attribut identifié par ce chemin doit être présent dans la structure de traits à l'issue de l'analyse. La seconde forme est une contrainte d'*inexistence* (préfixée par le symbole ˜) et spécifie que l'attribut doit être absent de la structure de traits.

Ainsi, par exemple, supposons que nous souhaitons ajouter la description des constructions avec pronominalisation clitique du complément d'objet direct dans notre grammaire du verbe *donner*. Nous pouvons représenter élégamment ces transformations en utilisant les contraintes existentielles.

La figure 4.27 présente le nouveau graphe V qui décrit le noyau verbal de

la phrase et dans lequel nous avons rajouté la possibilité de reconnaître un pronom clitique accusatif à gauche du verbe conjugué ; dans ce cas, nous unifions l'attribut *ppvacc* du non terminal V avec la forme fléchie du pronom clitique (équation *ppvac=$$.form*).

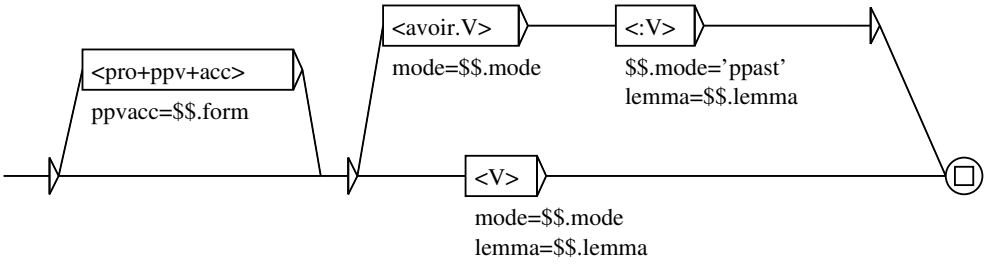

FIG. 4.27 – non terminal V modifié

Nous modifions également le graphe N0escroquerN1aN2 de manière à décrire ces nouvelles constructions (figure 4.28) : nous ajoutons en parallèle, à la reconnaissance de l'argument N1 en position d'objet direct, un nouveau chemin étiqueté en entrée par le mot vide contenant les deux équations :

n1=V.ppvacc
n1

La première équation unifie l'attribut *n1* associé au constituant P avec l'attribut *ppvacc* associé au noyau verbal de la phrase. Ce trait a pour valeur la forme fléchie du pronom clitique accusatif lorsqu'il est présent (figure 4.27) et a une valeur non spécifiée dans le cas contraire. La deuxième équation existentielle *n1* impose que le trait *n1* existe et que sa valeur soit spécifiée ; en d'autres termes, la seconde équation permet de s'assurer de la présence du pronom clitique dans la phrase. Nous procédons de la même manière pour décrire les *constructions alternées* où l'argument N2 prend la position d'objet direct.

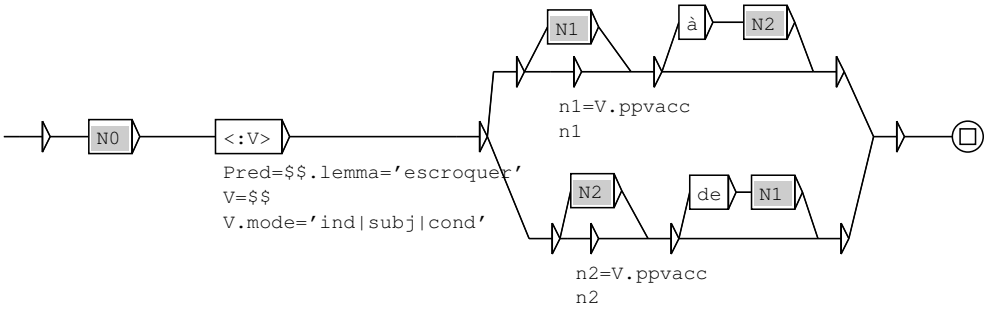

FIG. 4.28 – Nouveau graphe N0escroquerN1aN2

La grammaire ainsi modifiée analysera de nouvelles constructions telles que

Le fisc les a escroqués d'un sacré pactole
(ces 3 000 francs,) *Luc les a escroqués au fisc*

avec, comme résultat de l'analyse de la première phrase, l'arbre syntaxique et la structure de traits représentés par les figures suivantes :

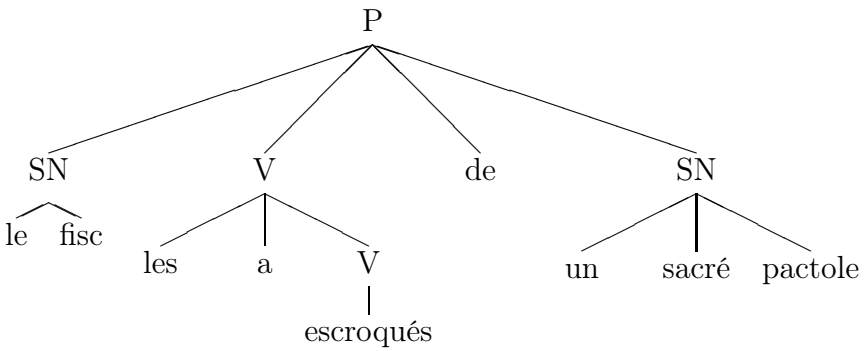

FIG. 4.29 – Arbre syntaxique pour la phrase *Le fisc les a escroqués d'un sacré pactole*

$$\begin{bmatrix} Pred : (1)escroquer \\ V : \begin{bmatrix} lemma : (1) \\ mode : ind \\ ppvacc : (2)les \end{bmatrix} \\ n0 : fisc \\ n1 : pactole \\ n2 : (2) \end{bmatrix}$$

FIG. 4.30 – Structure de traits obtenue suite à l'analyse de la phrase

Equation ensembliste

Les équations ensemblistes forment le troisième et dernier type d'équation pouvant décorer une grammaire DRTN. Ce type d'équation permet de construire des structures de traits à valeur ensembliste (cf. section 4.3.1), c'est-à-dire des attributs ayant pour valeur un ensemble de structures de traits. Une équation ensembliste à la forme suivante

$$chemin_1 < chemin_2$$

où $chemin_1$ et $chemin_2$ sont des chemins fonctionnels, et le trait désigné par $chemin_2$ a une valeur ensembliste. Une équation de cette forme permet de rajouter la valeur de l'attribut désigné par $chemin_1$ à l'ensemble de valeurs de l'attribut désigné par $chemin_2$.

Nous utilisons les équations ensemblistes pour construire des structures contenant un ensemble d'éléments dont on ne connaît pas a priori le nombre. Par exemple, la conjonction *et* permet de coordonner un ensemble de noms dont le nombre est a priori non borné au sein d'un même syntagme nominal :

Claudia et Takuya
Claudia, Mariana et Takuya
Claudia, Mariana, Lidia et Takuya
etc.

Nous pouvons facilement analyser ce type de constructions à l'aide des équations ensemblistes comme le montre le graphe de la figure 4.31.

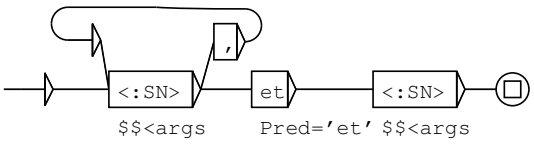

FIG. 4.31 – Représentation des SN coordonnés

Ce graphe analysera par exemple le syntagme nominal

Vincent, François, Paul et les autres

en produisant la structure de traits suivante :

$$\left[\begin{array}{l} Pred : et \\ args :< [\ head : Vincent\ ], [\ head : François\ ], [\ head : Paul\ ], [\ head : autres\ ] > \end{array}\right]$$

Raccourcis

Comme nous l'avons vu avec notre grammaire d'exemple, beaucoup des contraintes d'unification sont utilisées dans le but de faire remonter un attribut hérité d'un des sous-constituants au niveau du constituant englobant de manière à faire remonter des informations à des niveaux supérieurs dans l'arbre syntaxique. Ce type d'équation a toujours la même forme :

attribut=$$.attribut

(voir par exemple les équations *gender=$$.gender*, *number=$$.number* et *subcat=$$.subcat* dans le graphe SN de la figure 4.26).

Pour des raisons de facilités d'écriture et de lecture des grammaires, ainsi que de paresse, ce type d'équation peut être écrite de façon abrégée sous la forme suivante :

^attribut

Le graphe SN de la figure 4.26 peut être ainsi considérablement allégé en utilisant ce type de notation comme le montre la figure 4.32.

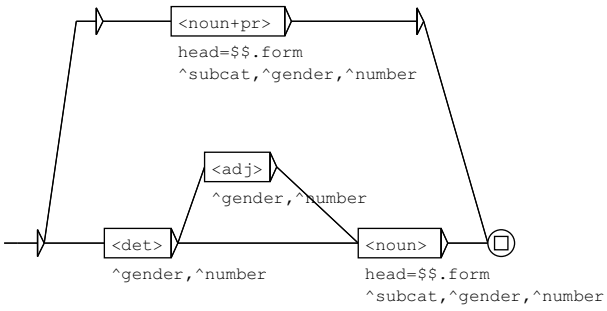

FIG. 4.32 – Utilisation des raccourcis d'écriture dans les contraintes d'unification

Enfin, l'équation ^ *$$* permet d'unifier dans sa totalité la structure de traits associée au symbole qui étiquette la boîte sous laquelle se trouve l'équation avec la structure de traits associée au constituant courant. Dans la pratique, nous n'avons pas utilisé ce type de contrainte durant l'écriture de notre grammaire. Cependant, les contraintes de cette forme peuvent certainement s'avérer utiles pour l'écriture d'autres grammaires dans le formalisme DRTN. Les grammaires LFG utilisent d'ailleurs intensivement des équations fonctionnelles à la sémantique équivalente (de la forme $\uparrow=\downarrow$).

4.4.1 Comparaison avec d'autres formalismes

Dans cette section, nous situons le formalisme des RTN décorés par rapport à d'autres formalismes grammaticaux existants qui ont été ou sont encore utilisés à des fins d'analyse syntaxique du français.

Grammaires locales

Les grammaires locales sont bien adaptées pour l'analyse de sous-langages spécifiques et pour l'analyse superficielle de textes en décrivant des phénomènes de micro-syntaxe (reconnaissance de groupes nominaux non récursifs, d'entités nommées, etc.). Dans [Paumier, 2003a], Sébastien Paumier montre

qu'il est également possible d'utiliser ce formalisme simple pour l'analyse syntaxique à large couverture. Cependant, ce travail s'est limité à l'analyse des phrases simples, et ne traite pas le problème des phrases enchassées (complétives, relatives, participiales, subordonnées circonstancielles, etc.). De plus, les grammaires locales ne permettent pas de formaliser de façon simple les phénomènes d'accords grammaticaux entre les constituants. Le travail présenté ici poursuit les recherches entreprises par S. Paumier. Tous ces problèmes non traités nous ont amené à enrichir le formalisme des grammaires locales par des contraintes d'unification afin de pallier toutes ces limitations.

LFG

Le formalisme des RTN décorés et celui de LFG sont très proches ; la principale différence étant qu'avec les RTN décorés, les règles de réécriture sont décrites sous la forme d'automates contrairement aux règles de réécriture hors-contexte utilisées dans LFG. L'utilisation d'automates nous permet de mettre aisément en relation (puisque dans le même graphe) différentes variantes syntaxiques considérées comme syntaxiquement équivalentes, telles que l'alternance sur la position des arguments par exemple.

Une autre différence fondamentale entre les deux approches est qu'avec LFG, les propriétés de sous-catégorisation et transformationnelles sont codées dans le lexique et non dans la grammaire. Ainsi, les règles de réécriture qui constituent une grammaire LFG sont typiquement générales et peu nombreuses ; la grammaticalité des phrases ainsi analysées est validée dans une seconde passe en testant la validité des structures fonctionnelles qui ont été construites à partir des stuctures qui sont associées à chaque élément du lexique présent dans la phrase à partir de règles de bonnne formation. Notre approche est différente dans le sens où notre grammaire est fortement lexicalisée et que nous décrivons explicitement, pour chaque élément prédicatif, quelles sont les possibles réalisations syntaxiques des constituants dans lesquels ils peuvent apparaitre.

Enfin, le formalisme LFG est avant tout un formalisme linguistique, dans lequel les structures de traits doivent être sous une certaine forme pour être considérées comme bien formées. Le formalisme de RTN décorés, en contre-

partie, est avant tout un outil formel qui ne spécifie aucune contrainte sur la constitution des structures de traits. Nous nous en sommes servi à des fins d'analyse syntaxique, mais ce même formalisme peut être utilisé pour d'autres types d'applications. Il a d'ailleurs déjà été utilisé à des fins d'extraction d'information [Watrin, 2006].

TAG

Le modèle des grammaires d'arbres adjoints (ou TAG) a été initialement défini dans [Joshi *et al.*, 1975], puis dans une version plus récente intégrant les structures de traits et l'unification dans [Vijay-Shanker, 1987]. Anne Abeillé a entrepris, dans le cadre de sa thèse [Abeillé, 1991], la construction d'une grammaire d'arbres adjoints pour le français en utilisant les ressources linguistiques élaborées aux LADL. Ce travail de description continue depuis plus de 15 ans pour donner lieu à une grammaire à large couverture d'une finesse de précision pour de nombreux phénomènes syntaxiques [Abeillé, 2002], qui est encore enrichie aujourd'hui [Barrier, 2006]. Les grammaires TAG appartiennent à la famille des grammaire d'unification [Abeillé, 1993] mais ce distinguent des autres grammaires de ce type en ce que les unités sont des arbres et non des règles de réécriture hors-contexte. Ces arbres se combinent entre eux à l'aide d'opérations de substitution et d'adjonction.

Le résultat de l'analyse d'un énoncé par une grammaire TAG se présente sous la forme d'un arbre dérivé et d'un arbre de dérivation. Ce dernier fournit une représentation sémantique des énoncés analysés, dans lequel apparaissent les relations prédicat-argument (obtenus par substitution) et modifieur-modifié (obtenu par adjonction). La forme d'un arbre pour représenter les résultat d'analyse est plus limitée qu'une représentation sous la forme d'une structure de traits puisqu'elle ne permet pas de représenter les relations de coréférences (par exemple lorsque un même complément est l'argument de plusieurs prédicats), ce qui peut se représenter à l'aide de structure de traits à valeur partagée.

De plus, l'adjonction permet de représenter de manière simple les verbes à complétive et les dépendances à distance, cependant, avec ce type d'analyse, les verbes à complétive sont considérés comme modifieurs de la phrase

subordonnée.

Ainsi, si nous considérons les séquences suivantes :

Jean pense que Marie partira
Jean pense partir
Jean pense à son départ

Ces trois séquences recevront les arbres de dérivations suivants à la suite de leur analyse par la grammaire TAG du français telle que présentée dans [Abeillé, 2002] [1] :

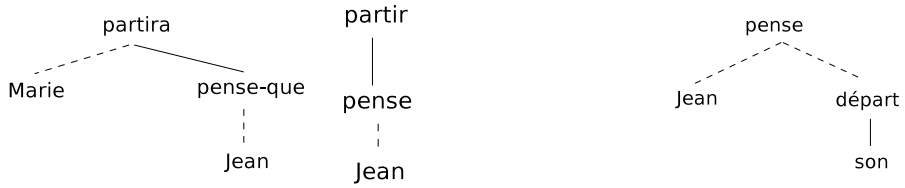

FIG. 4.33 – Arbres de dérivation obtenus après l'analyse TAG

Avec cette analyse, lorsque le complément du verbe *penser* a la forme d'une proposition complétive ou infinitive, ce verbe est considéré comme un modifieur de la proposition subordonnée, ce qui n'est pas le cas lorsque son complément à la forme d'un syntagme nominal. Nous ne sommes pas vraiment en accord avec ce type d'analyse (qui a néanmoins l'avantage d'être commode, puisque l'utilisation d'arbres auxiliaires pour représenter les verbes à compléments phrastiques permet de formaliser de façon très simple les phénomènes d'extraction et les dépendances à distance). En effet, nous pensons que le verbe principal *penser* entretient le même type de relation avec ces deux arguments dans les trois phrases de notre exemple. C'est pourquoi nous associons à ces trois séquences, des analyses dans lesquelles ce verbe est considéré comme le prédicat principal à deux arguments, le second ayant la forme d'une complétive, d'une infinitive ou d'un syntagme nominal (cf. figure 4.34).

[1] Les arcs en pointillés représentent les relations entre un prédicat et son argument (obtenus par subsitution) et les arcs plein les relations entre un modifié et son modifieur (obtenu par adjonction).