Institut d'électronique et d'informatique Gaspard-Monge

ANALYSE SYNTAXIQUE à partir des tables du Lexique-Grammaire

Elsa Tolone

Journée des doctorants 20 janvier 2009

INTRODUCTION

- Domaine: Analyse syntaxique

  = Construire la structure grammaticale d'une phrase

  → Difficulté: Complexité et grandeur de la langue

- Applications:
  - Compréhension de texte
  - Extraction d’information
  - Traduction

- Analyseurs:
  - symboliques = grammaire + lexique
  - probabilistes = modèle acquis sur corpus annoté manuellement

CONTEXTE

- Tables du Lexique-Grammaire
  - Développées depuis 30 ans par l'Equipe d'Informatique Linguistique de l'IGM
  - Actuellement: Projet LGTag
- Caractéristiques:
  - Décrivent caractéristiques grammaticales des mots
  - Ressources lexicales riches
  - Pas utilisables directement dans un analyseur symbolique

OBJECTIFS

- Rendre les tables du Lexique-Grammaire exploitables dans un analyseur syntaxique
- Développer et évaluer un analyseur syntaxique qui exploite les tables du Lexique-Grammaire

Plan

1. Tables du Lexique-Grammaire
2. Analyse TAG avec DyALog
3. Formalisation des tables du Lexique-Grammaire
4. Intégration à un analyseur TAG
5. Conclusion et perspectives

Tables du Lexique-grammaire

Historique

- M. Gross, 1975
- Étude de la phrase élémentaire (ou cadre de sous-catégorisation)

  ex: *N0 V N1*
- Usages des verbes, noms, adjectifs ou adverbes prédicatifs + expressions figées

  → certaines propriétés en commun
- Les différents emplois sont distingués (ex: cuisiner)
- 13,400 verbes dans 60 tables

Principe

- Codage dans une table:

  lignes = entrées (mots) , colonnes = propriétés
- Propriétés: constructions, transformations, distributions

  ex: N0 := Nhum → noms de personnes
- Chaque propriété grammaticale testée pour chaque entrée
- Codage binaire:

  '+': propriété acceptée

  '-': non
- Codage lexical

  ex: prépositions
- Constructions définitoires pour chaque table

  ex pour la table V_33: *N0 V à N1*

Exemple

N0 =: Nhum	N0 =: N-hum	N0 =: Nnr	N0 être V-n	Ppv	Ppv =: se figé	Ppv =: en figé	Ppv =: les figé	Nég	<ENT>	N0 V	N0 être V-ant	N0 est Vpp	N0 V de N0pc	N1 =: Nhum	N1 =: N-hum	N1 =: le fait Qu P	Ppv =: lui	Ppv =: y	N1 être V-n	N0hum V W sur ce point	[extrap]	N0idée V Loc N1esprit	<OPT>
+	-	-	-	<E>	-	-	-	-	accrocher	+	-	+	-	+	+	-	-	+	-	+	+	-	Paul accroche à ses nouvelles fonctions
~	~	~	~	s'	+	-	-	-	acliquer	~	~	~	~	~	~	~	~	~	~	~	~	~	Paula s'est acliqué (à+avec)Paulo
+	-	-	-	<E>	-	-	-	-	adhérer	-	-	-	-	-	+	-	-	+	-	-	+	-	Max adhère à(deux partis+un pacte)
-	+	+	-	s'	+	-	-	-	adresser	-	-	-	-	+	-	-	-	+	-	-	-	-	Cette mesure s'adresse aux enfants
+	-	-	-	s'	+	-	-	-	adresser	-	-	-	-	+	-	-	-	+	-	-	-	-	Paul s'adresse à Max

Extrait de la table des verbes de la classe 33

Caractéristiques

- Avantages:
  - Description riche
  - Large couverture
  - Base linguistique solide

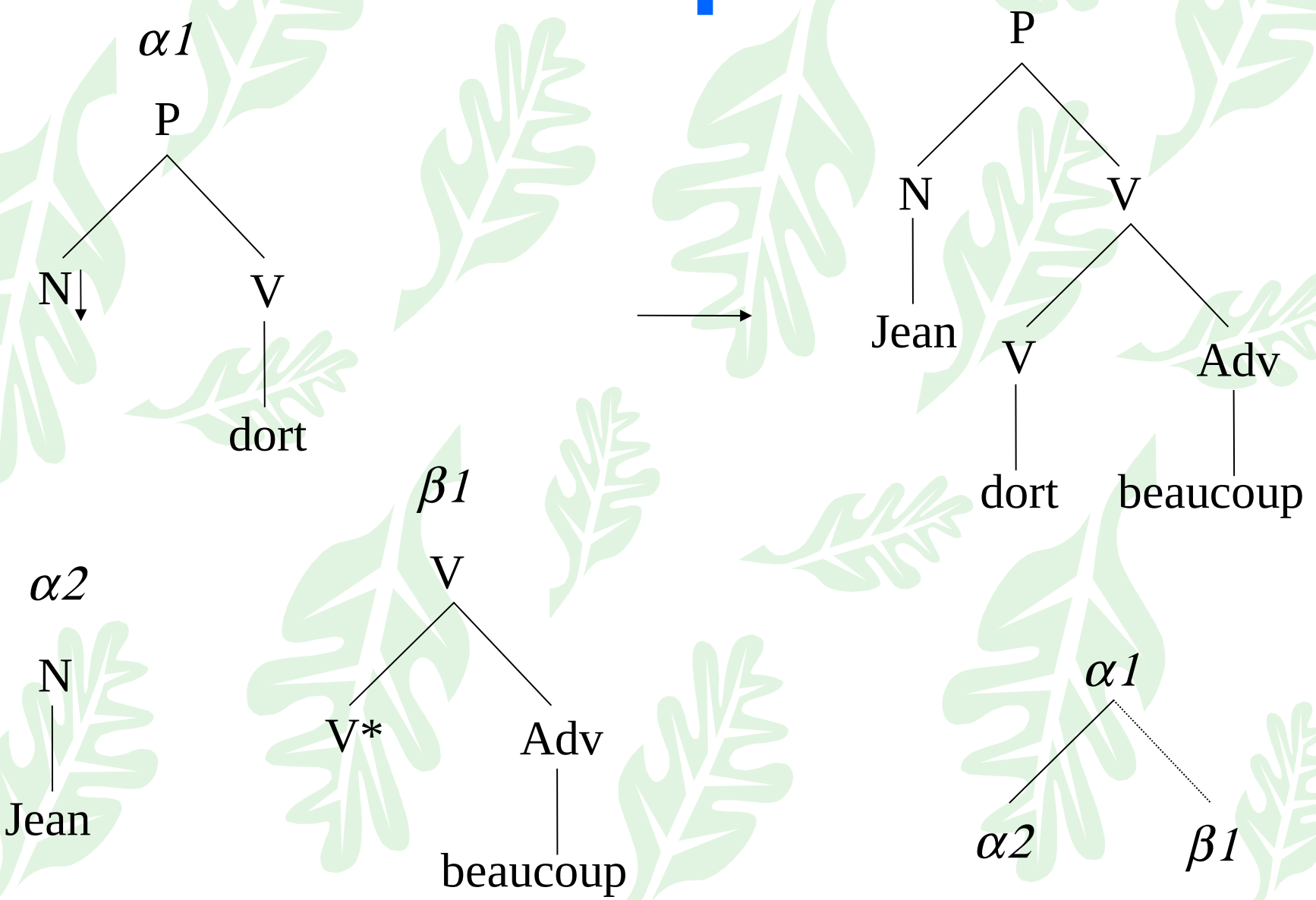
- Inconvénients:
  - Informations implicites dont les plus importantes
  - Incomplétude des tables du lexique-grammaire
  - Manque de formalisation

Analyse TAG avec DyALog

Grammaire d'arbres adjoints

- Joshi, 1975
- Ensemble fini d'arbres élémentaires de 2 types :
  - arbres initiaux (α) → opération de substitution
  - arbres auxiliaires (β) → opération d'adjonction
- Combinaison de plusieurs arbres élémentaires :
  - arbre dérivé (resultat de la combinaison)
  - arbre de dérivation (ensemble des opérations qui représentent l'arbre dérivé)

Exemple


α1
P
N↓
V
dort
P
N
V
Jean
V
Adv
dort
beaucoup
β1
V
V*
Adv
beaucoup
α2
N
Jean
α1
α2
β1

Métagrammaire

- Chaque classe décrit une notion linguistique

  ex: sujet canonique

  → Description partielle d'un arbre TAG

  + contrainte de combinaison avec les autres classes

- L'ensemble des classes forment un arbre TAG

- Niveau plus abstrait de représentation

  → permet de régler le problème de l'explosion du nombre des arbres TAG

DyALog

- Developpé par Eric de le Clergerie
- Environnement de programmation logique adapté pour le TAL qui inclus :
  - un formalisme développé de métagrammaires
  - un outil pour les compiler en grammaire TAG
- Permet la construction d'un analyseurs syntaxique efficace à partir de:
  - une grammaire TAG
  - un lexique syntaxique qui spécifie le comportement grammatical de chaque mot de la langue
- FRMG: métagrammaire et analyseur syntaxique

Formalisation des tables du Lexique-Grammaire

Travail en cours pour les verbes, noms et expressions figées:

- Noms différents pour une même propriété

  → Harmonisation des intitulés de colonnes

- Constructions définitoires implicites (littérature)

  → Constante '+' ou '-' pour l'ensemble de la table

- Toutes les propriétés ne sont pas codées dans chaque table

  → Codage '+', '-' ou 'o' pour l'ensemble de la table

- Intitulés des propriétés pas entièrement formalisés

  → Tables converties en un lexique formalisé

Table des classes

table	N0 =: Nhum	N0 =: N-hum	N0 =: Nnc	N0 =: Nnr	N0 =: V1-inf W	<ENT>	Ppy =: se figé	N0 V	N0 V N1	zone 1	N0 V à N1	N1 =: Nhum	N1 =: N-hum	N1 =: Qu P	N1 =: Qu Psubj	N0 V Prep N1 V0-inf W	N0 V N1 V0-inf W	N0 V V0-inf W	N0 U prep V0-inf	N0 U Prep Nhum	N0 U Prep N-hum	N0 U Nhum	N0 U N-hum
V_1	o	-	o	-	-	o	o	-	-	o	-	-	-	-	-	-	-	-	+	o	o	o	o
V_2	+	-	-	-	-	o	o	-	-	-	-	-	+	-	-	o	o	+	-	-	-	-	-
V_4	-	-	-	+	+	o	-	o	+	-	-	o	o	-	-	-	-	-	-	-	-	-	-
V_31R	o	o	-	-	-	o	o	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V_31H	+	-	-	-	-	o	o	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V_33	o	o	-	o	-	o	o	o	-	-	+	o	o	-	-	-	-	-	-	-	-	-	-
V_32H	o	-	-	o	-	o	o	-	+	-	-	+	-	-	-	-	-	-	-	-	-	-	-

Extrait de la table des classes des verbes

LGExtract

Outil qui transforme les tables en format xml ou texte semi-structuré (Constant & Tolone 2008)

ID=V_33_8
lexical-info=[cat="*verb*",
verb=[lemma="*adresser*",ppvse="*true*"]]
args=(const=[pos="*0*",
dist=(comp=[cat="*NP*",hum="*true*",
origine=(orig="*N0 =: Nhum*")])],
const=[pos="*1*",
dist=(comp=[cat="*NP*",hum="*true*",
origine=(orig="*N1 =: Nhum*")])])
all-constructions=[absolute=(construction="*N0 V à N1*"),
relative=(construction="*Ppv =: y*")]
example=[example="*Paul s'adresse à Max*"]

→ Il faut interpréter un certains nombre de colonnes

Conversion Iglex → Lefff

Développé par Benoît Sagot pour son lexique syntaxique Lefff

→ Format entièrement formalisé qui permet d'être utilisé dans les analyseurs, notamment FRMG

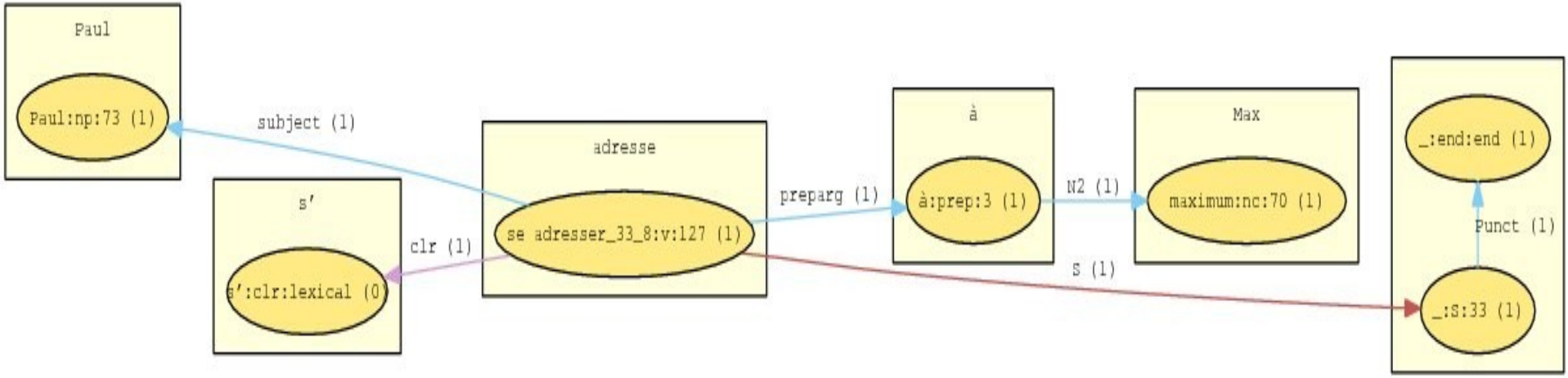
adresser___33_8 v-er:std 100;s'Lemma;v;<Suj:cln|sn,Objà:à-sn|y>;cat=v,@pron,@SujNhum,@ObjàNhum;%actif

- fonctions syntaxiques: Suj, Objà ...
- réalisations: cln, sn, à-sn, y ...
- macros: @pron, @SujNhum, @ObjàNhum ...
- redistributions: %actif, %actif_impersonnel, %passif ...

Intégration à un analyseur TAG

- Actuellement:
  - Métagrammaire FRMG
  - Verbes issus du Lexique-Grammaire
    + autres familles de mots du Le*fff*
  - → Analyseur qui repose sur les verbes du Lexique-Grammaire
- A terme:
  - Intégrer toutes les catégories
  - Evaluer analyseur syntaxique obtenu:
    - Temps d'analyse
    - Degré d'ambiguïté
    - Précision

Exemple



Paul s'adresse à Max

CONCLUSION

- Objectif thèse

  → Intégrer les tables du Lexique-Grammaire dans un analyseur syntaxique

- Difficultés lors de la conversion:
  - Informations manquantes
  - Formalisation des constructions

- Sujet très riche

  → Interface entre linguistique et informatique

PERSPECTIVES

- Richesse des tables du Lexique-Grammaire

  → Métagrammaire FRMG à compléter, étendre ou adapter

- Utilisation de l'analyseur sur un gros corpus

  → Pondérer les cases de façon moins binaire

- Tables pour d'autres langues

  → Même méthodologie pour construire des analyseurs pour d'autres langues