Méta-grammaire et calcul sémantique pour les grammaires d'arbres adjoints

Yannick Parmentier

yannick.parmentier@loria.fr

Soutenance de Stage ST50 / DEA IAP

Université de Technologie de Belfort Montbéliard

Université de Franche-Comté

Introduction

- **Lieu du stage** : Equipe Langue et Dialogue - LORIA
- **Domaine** : Traitement Automatique du Langage Naturel
- **Objectif** : réaliser l'écriture semi-automatique d'une grammaire d'arbres adjoints à portée sémantique

Plan

1. Présentation des Grammaires d’Arbres Adjoints (TAG)

Plan

1. Présentation des Grammaires d'Arbres Adjoints (TAG)

2. Ajout d'une dimension sémantique dans les grammaires TAG

Plan

1. Présentation des Grammaires d'Arbres Adjoints (TAG)
2. Ajout d'une dimension sémantique dans les grammaires TAG
3. Ecriture semi-automatique d'une grammaire TAG : notions de « méta-grammaire » et « compilateur »

Plan

1. Présentation des Grammaires d'Arbres Adjoints (TAG)

2. Ajout d'une dimension sémantique dans les grammaires TAG

3. Ecriture semi-automatique d'une grammaire TAG : notions de « méta-grammaire » et « compilateur »

4. Deux implémentations de compilateurs : MGC et résolveur de descriptions colorées

Plan

1. Présentation des Grammaires d'Arbres Adjoints (TAG)

2. Ajout d'une dimension sémantique dans les grammaires TAG

3. Ecriture semi-automatique d'une grammaire TAG : notions de « méta-grammaire » et « compilateur »

4. Deux implémentations de compilateurs : MGC et résolveur de descriptions colorées

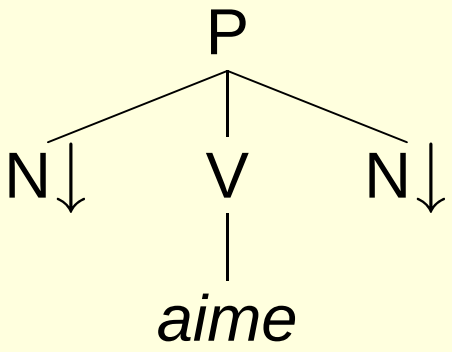
5. Conclusion et perspectives

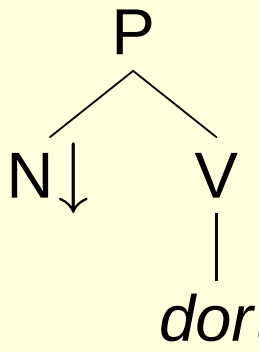
1. Présentation des Grammaires d'Arbres Adjoints

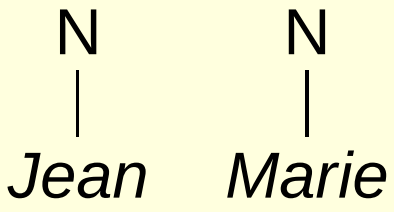
- Définition formelle d'une grammaire TAG : $G = (NT, T, P, I, A)$
- I et A : ensembles d'arbres ayant des caractéristiques spécifiques
- Deux opérations de combinaison d'arbres : *adjonction* et *substitution*
- Système de réécriture d'arbres $\rightarrow$ processus de dérivation

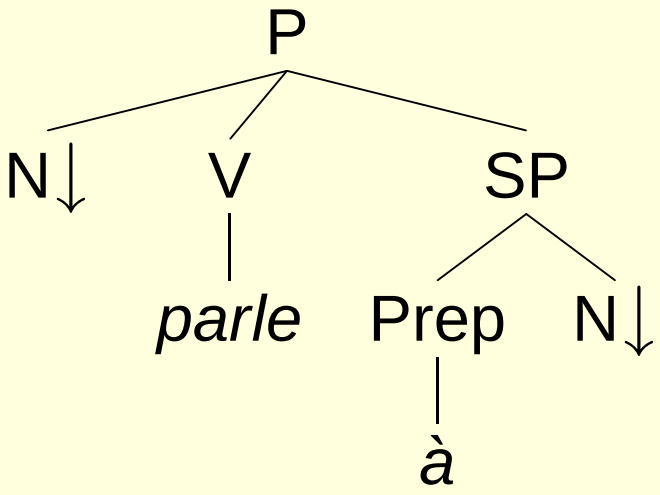
1. Présentation des Grammaires d'Arbres Adjoints

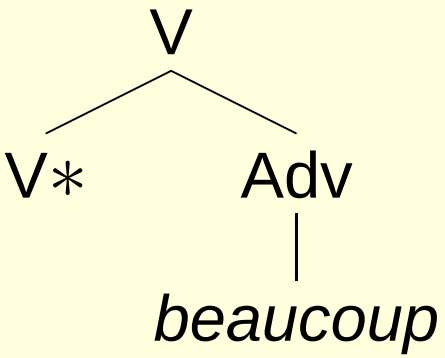
Exemple d'unités élémentaires d'une grammaire TAG :





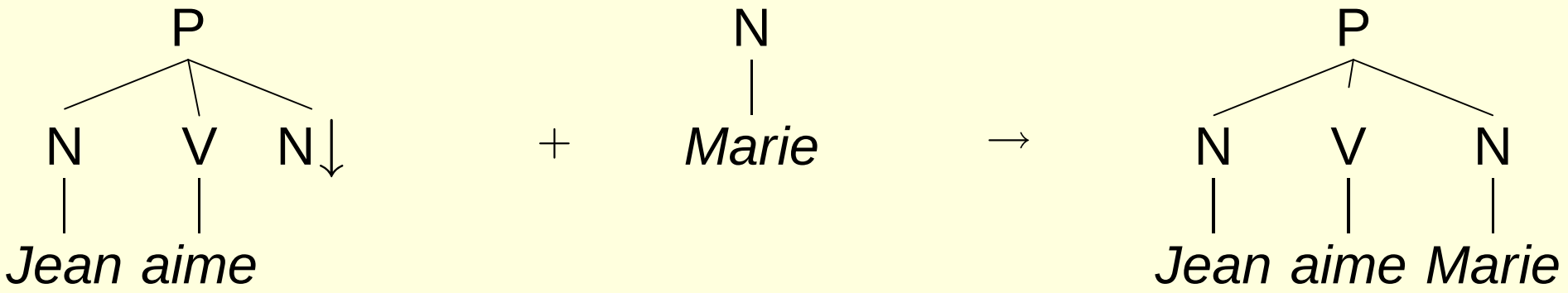






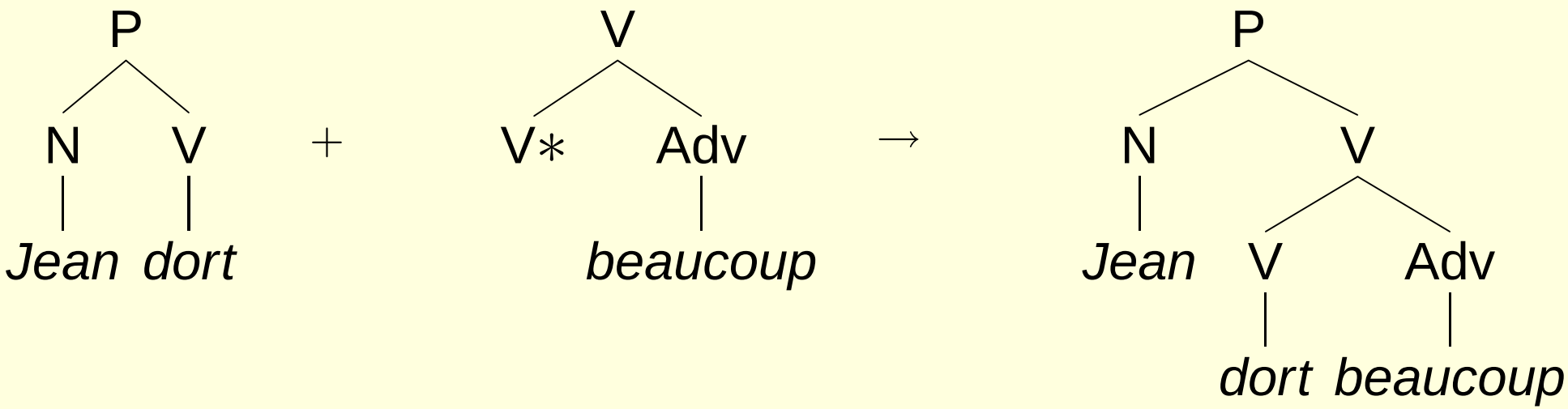
1. Présentation des Grammaires d'Arbres Adjoints

Exemple de substitution :



1. Présentation des Grammaires d'Arbres Adjoints

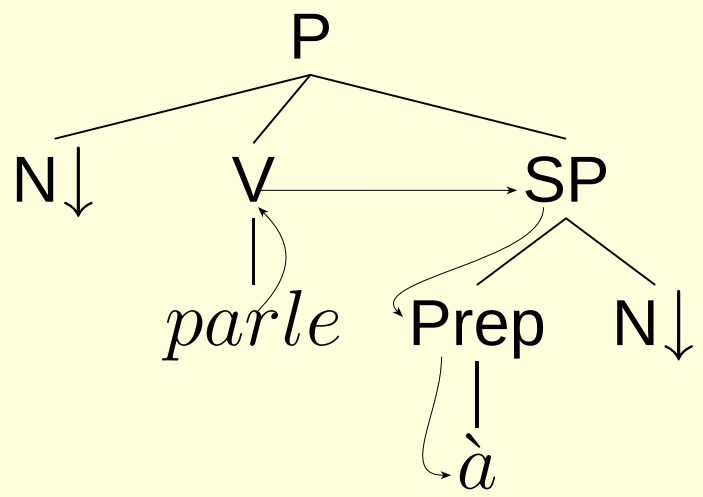
Exemple d'adjonction :



1. Présentation des grammaires d'arbres adjoints

Propriétés linguistiques et formelles :

- Domaine de localité étendu (par rapport aux grammaires hors-contexte)
  exemple :



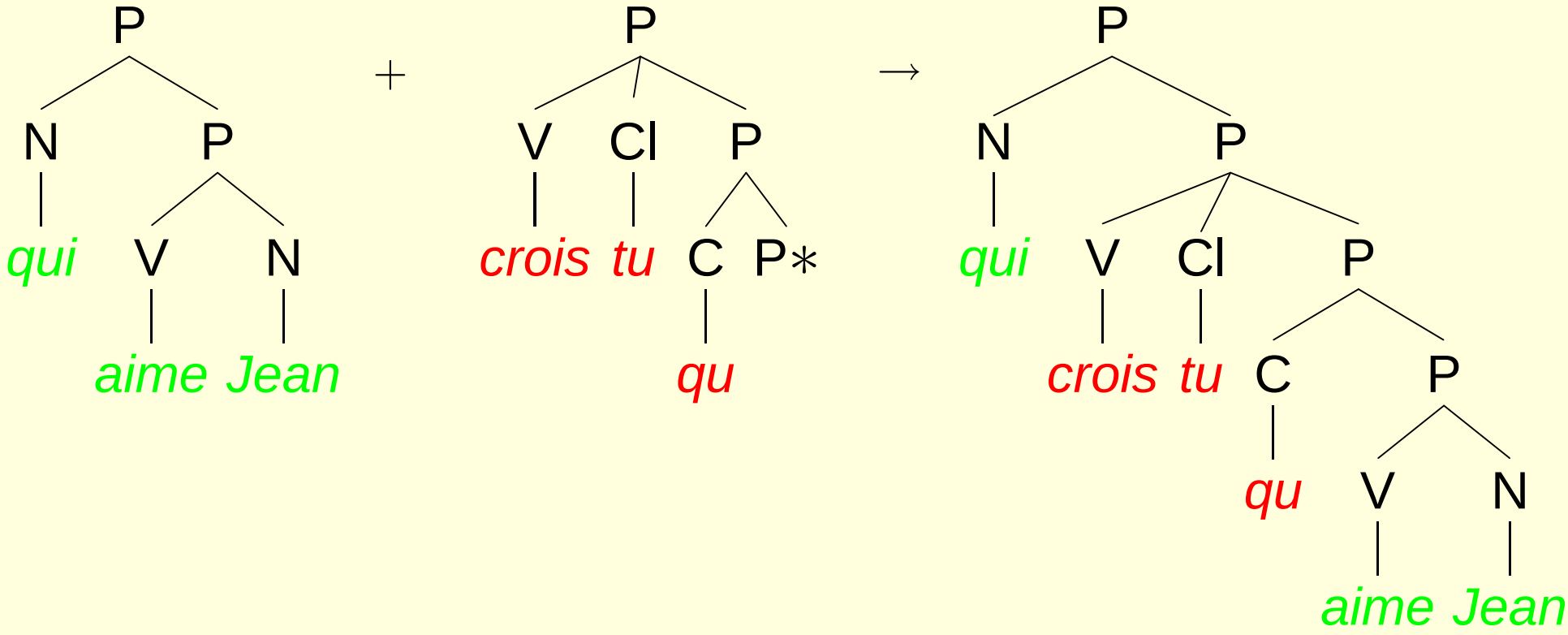
(TAG)

$$\left\{\begin{array}{llll} P & \rightarrow & N\ V\ SP & + \\ V & \rightarrow & parle & + \\ SP & \rightarrow & Prep\ N & + \\ Prep & \rightarrow & \grave{a} & \end{array}\right.$$

(hors-contexte)

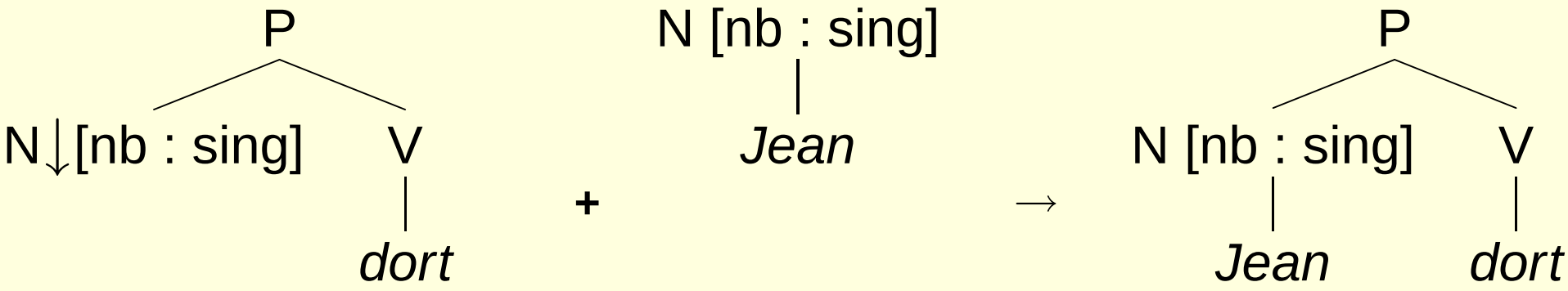
1. Présentation des grammaires d'arbres adjoints

- Factorisation de la récursion hors du domaine de dépendances (grâce à l'adjonction)
  exemple :



1. Présentation des grammaires d'arbres adjoints

Grammaires TAG à structures de traits (FTAG) :



$$\begin{bmatrix} a_1 : v_1 \\ a_2 : \mathbf{y} \\ a_3 : c \end{bmatrix} \cup \begin{bmatrix} a_1 : \mathbf{x} \\ a_2 : v_2 \\ a_3 : c \end{bmatrix} \Rightarrow \{\mathbf{x} = v_1 \wedge \mathbf{y} = v_2\}$$

2. Ajout d'une dimension sémantique dans les grammaires TAG

- motivations pour cet ajout :
  - utilisation des grammaires TAG en analyse et génération
  - syntaxe comme support de la représentation sémantique

- représentation sémantique $\rightarrow$ formule logique

- plusieurs types de logiques utilisables

- en TAG : sémantique *plate* (logique des prédicats « débranchée » + variables d'unification - cf [Gardent et Kallmeyer, 2003])

exemple :

« un chien aboie »

$$l_0 : \exists(x_1, l_1, l_2) \ \wedge \ l_1 : Chien(x_1) \ \wedge \ l_2 : Aboyer(x_1)$$

$$\equiv \ \exists x(Chien(x) \wedge Aboyer(x))$$

2. Ajout d'une dimension sémantique dans les grammaires TAG

Principe du calcul sémantique :

1. à chaque arbre de la grammaire, on associe une formule en sémantique plate

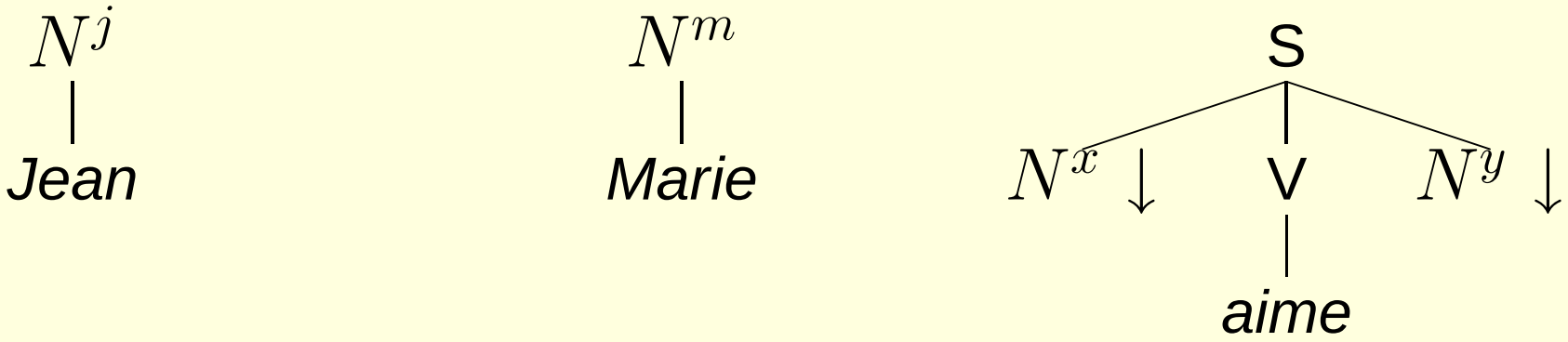
2. cette formule peut contenir des variables d'unification

3. on associe à certains nœuds des traits qui représentent des index sémantiques

4. ces index et les variables de la formule sont partagés par coindexation

2. Ajout d'une dimension sémantique dans les grammaires TAG

Composition des formules élémentaires :



$l_0 : nom(j, Jean)$ $l_1 : nom(m, Marie)$ $l_2 : Aimer(x, y)$

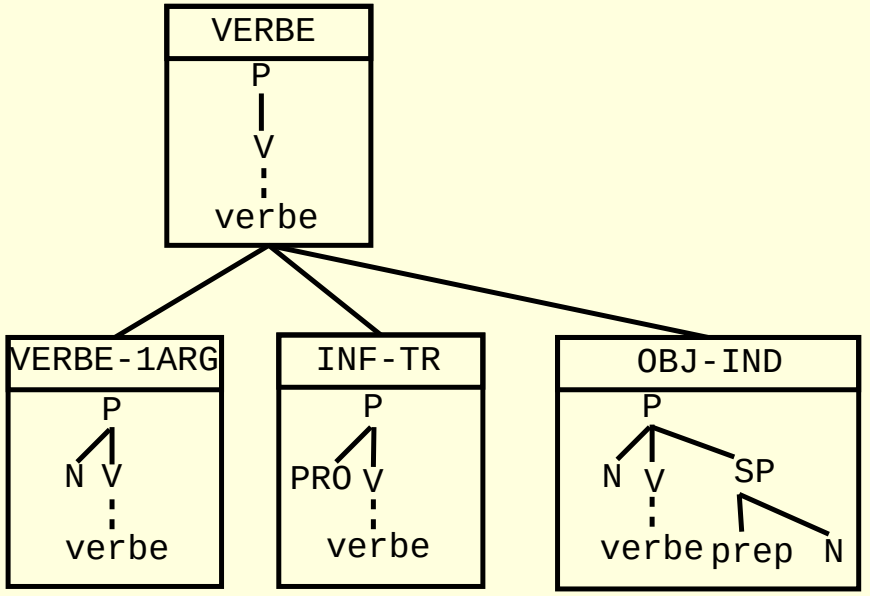
Formule après substitutions :

$$l_0 : nom(j, Jean) \wedge l_1 : nom(m, Marie) \wedge l_2 : Aimer(j, m)$$

3. Ecriture semi-automatique d'une grammaire TAG

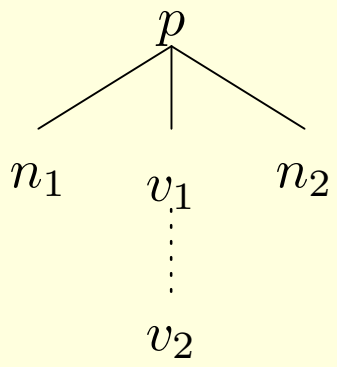
- constat : beaucoup d'arbres en partie identiques
- problème : comment factoriser l'information contenue dans les arbres ?
- solution proposée : placer les arbres (sans ancre lexicale) dans une hiérarchie d'héritage

exemple :



3. Ecriture semi-automatique d'une grammaire TAG

- dans chaque classe de notre hiérarchie : description partielle de la structure d'un arbre au moyen d'une logique de description d'arbres
- utilisation d'un langage utilisant les relations entre nœuds suivantes :
  $\lhd*$ (dominance large), $\lhd$ (dominance stricte), $\prec$ (précédence), $\approx$ (égalité).
- exemple :



$$(p \lhd n_1) \wedge (p \lhd v_1) \wedge (n_1 \prec v_1) \wedge (v_1 \lhd * v_2) \wedge (p \lhd n_2) \wedge (v_1 \prec n_2)$$

3. Ecriture semi-automatique d'une grammaire TAG

- cette hiérarchie d'héritage constitue ce que l'on appelle la **méta-grammaire**

- pour générer une grammaire à partir d'une méta-grammaire, on effectue des croisements entre classes terminales et on calcule le *référent minimal*, but : avoir des arbres élémentaires

- ce mécanisme est réalisé de manière automatique par un outil appelé **compilateur**

- attention : on souhaite encore interfacer la syntaxe (arbres) et la sémantique (formules) dans la méta-grammaire

4. Deux implémentations de compilateurs

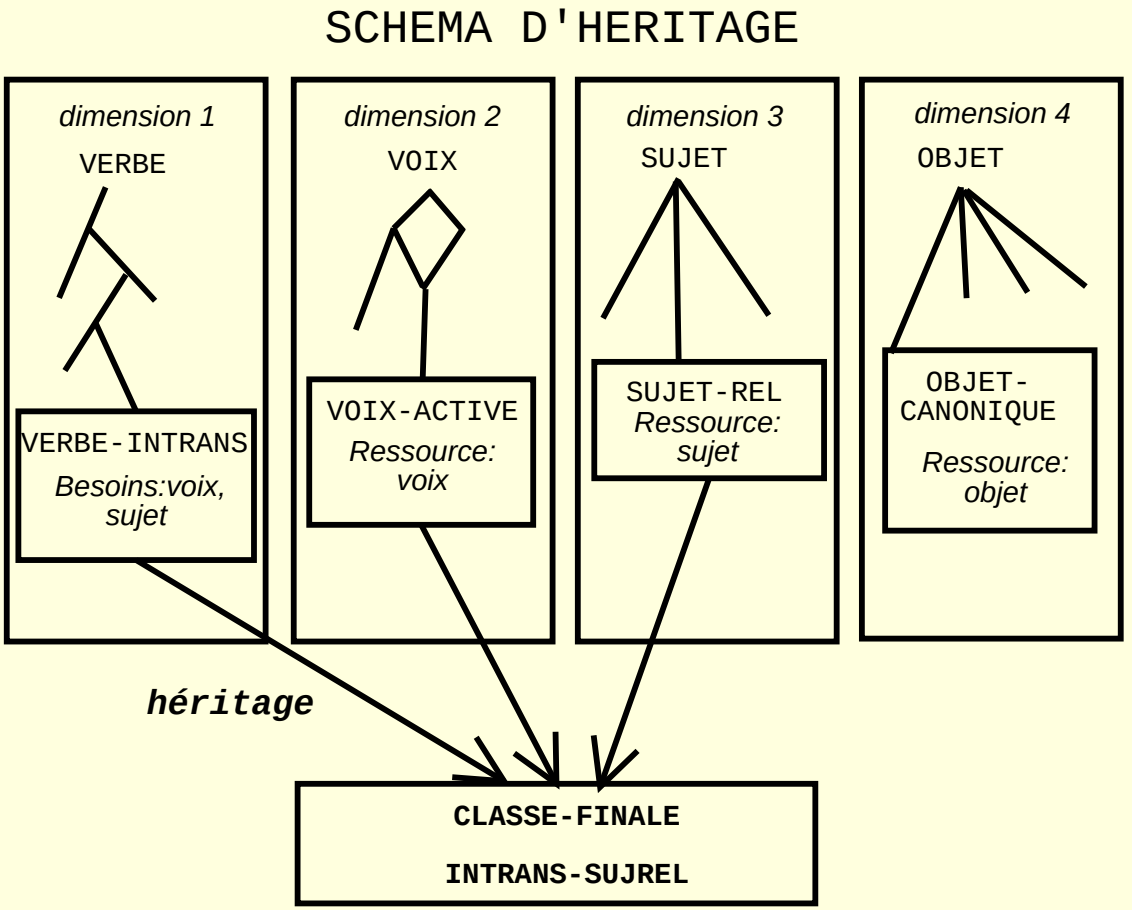
« Cœur » du stage : réaliser l'intégration de la sémantique dans les compilateurs développés dans l'équipe

1. Editeur/compilateur « MGC » : outil écrit en java générant des arbres

- Principe : la méta-grammaire décrit une hiérarchie de classes multi-dimensionnelle dans laquelle
  - les nœuds sont désignés par des constantes
  - les croisements sont guidés par des besoins et ressources

4. Deux implémentations de compilateurs : MGC

exemple :



4. Deux implémentations de compilateurs : MGC

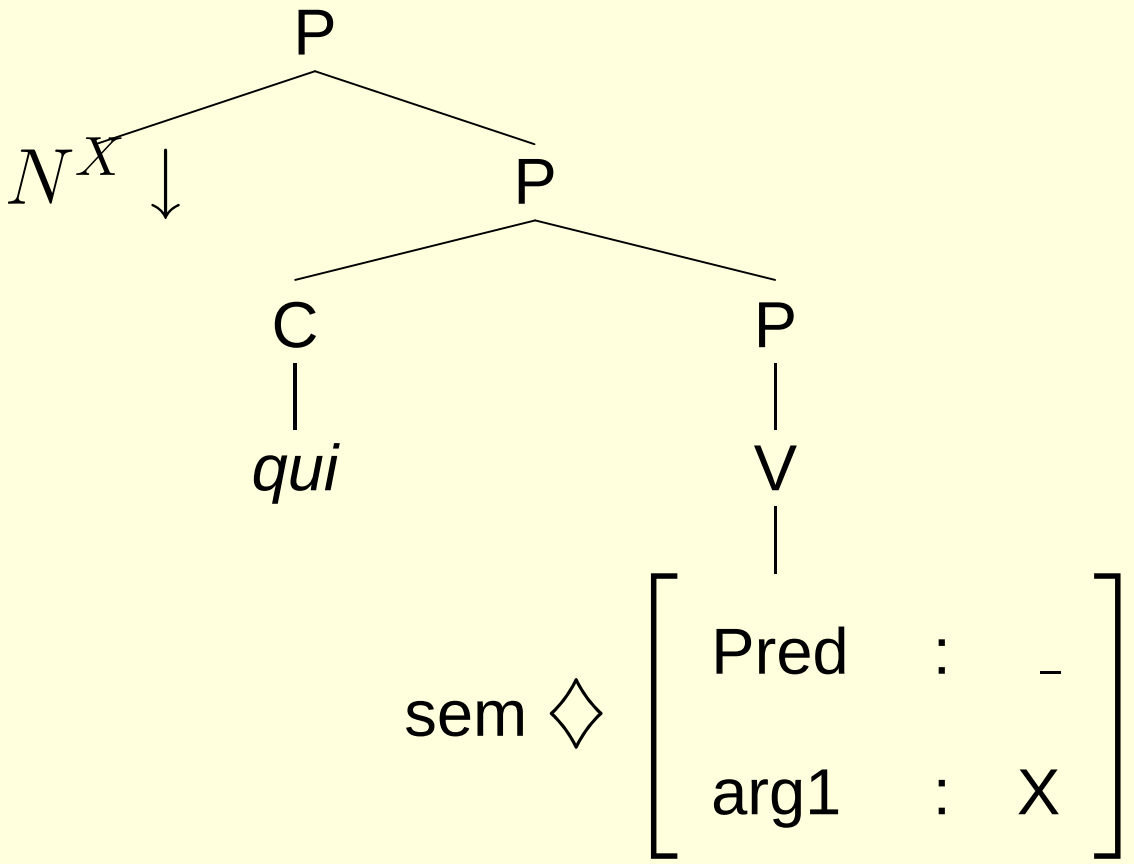
- Intégration de la sémantique : où placer la formule logique ?
  - sur un nœud dédié dans l'arbre (nœud « sémantique » défini dans notre hiérarchie)

- Résultats :
  - écriture d'une méta-grammaire à portée sémantique « jouet »,
  - construction de la représentation sémantique réussie lors de l'analyse

- Inconvénient : les nœuds des descriptions sont nommés par des constantes globales

4. Deux implémentations de compilateurs : MGC

exemple de la classe INTRANS-SUJREL :



(Le garçon qui dort)

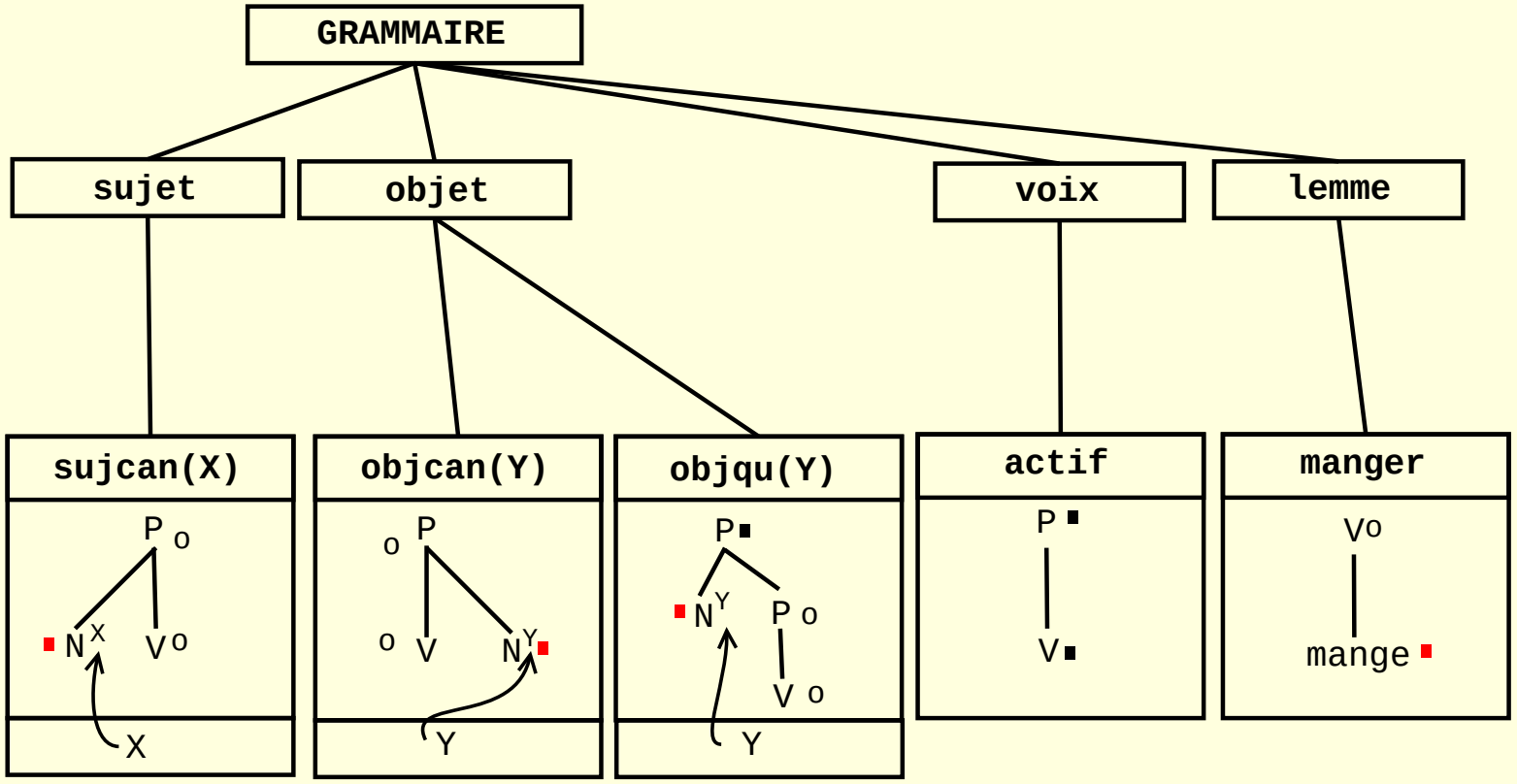
4. Deux implémentations de compilateurs : Résolveur

2. Résolveur de descriptions colorées : logiciel écrit en oz-mozart, et générant des arbres élémentaires (avec ancre)

- Principe : on effectue une factorisation plus forte dans la méta-grammaire en décrivant des fragments d'arbres
  - où les nœuds comportent une couleur parmi {rouge, noir, blanc}
  - qui vont être combinés par idenfication des nœuds colorés (suivant un code de couleurs)
- Avantage : nœuds « anonymes » (plus de problème de constantes globales)

4. Deux implémentations de compilateurs : Résolveur

exemple :



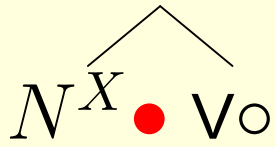
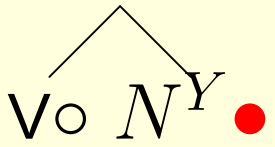
4. Deux implémentations de compilateurs : Résolveur

- Intégration de la sémantique : difficulté → comment repérer les nœuds contenant des traits à coindexer avec la formule ?
  - utilisation de classes à dimension syntaxique et/ou sémantique
  - « boîte de nommage » permettant la désignation locale d'un trait d'un nœud
  - descriptions explicites des combinaisons de classes avec coréférences

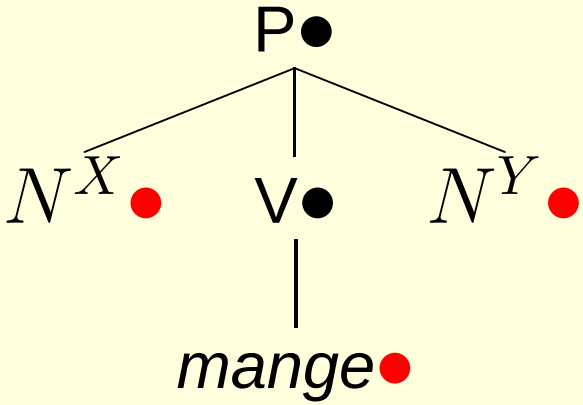
    exemple : $Manger ::=$
    $[sujcan(X) \wedge objcan(Y) \wedge actif \wedge manger \wedge semManger(X,Y)]$ |
    $[sujcan(X) \wedge objqu(Y) \wedge actif \wedge manger \wedge semManger(X,Y)]$ |
    $[sujcan(X) \wedge objcan(Y) \wedge passif \wedge manger \wedge semManger(Y,X)]$
    | ...
- Résultat :
  - génération d'une grammaire TAG de taille relativement importante intégrant une portée sémantique

4. Deux implémentations de compilateurs : Résolveur

Exemple de la combinaison $[sujcan(X) \wedge objcan(Y) \wedge actif \wedge manger \wedge semManger(X,Y)]$:

sujcan(X)	actif	objcan(Y)
P○ 	P●  V●	P○ 
	manger	semManger(X,Y)
	V○ *mange*●	mange(X, Y)

(arbre élémentaire)



mange(X,Y)

5. Conclusion et perspectives

- l'utilisation de la sémantique plate pour générer des grammaires TAG à portée sémantique a été implémentée avec succès dans les compilateurs utilisés à l'heure actuelle,

- 1er travail avec le compilateur MGC : validation de la théorie (sémantique plate appliquée à TAG), mais problème pour passer à une grammaire de taille importante

- 2nd travail avec le résolveur de descriptions : approche innovante basée sur des techniques éprouvées pour l'analyse (nœuds colorés), évite le problème du nommage des nœuds dans la méta-grammaire,

- la suite des travaux va concerner (1) l'interfaçage du résolveur avec d'autres outils (analyseurs, générateurs) puis (2) la validation des grammaires générées en analyse et en génération.