

# LGExtract: un outil générique pour reformater les tables du lexique-grammaire

*Matthieu Constant  
Université Paris-Est*

*le 21 janvier 2008  
Séminaire interne  
Equipe d'informatique linguistique  
Laboratoire d'informatique de l'IGM*

# Introduction

- Objectif : développer un outil générique de reformatage des tables de lexique-grammaire
- Applications :
  - Création d'un lexique syntaxique simplifié pour distribution
  - Création d'un lexique syntaxique pour analyse syntaxique automatique

# Lexique-grammaire - 1

- But : décrire de manière systématique le comportement syntaxique des prédicats de la langue
- Prédicats :
  - Verbes (ex. Luc **mange** une pomme)
  - Noms (ex. Max fait une **colère**)
  - Adjectifs (ex. Marie est **consciente** de son charme)
  - Phrases figées (ex. Lea **perd la tête**)

## Lexique-grammaire - 2

- Les prédicats sont divisés en classes sur des critères formels
- Exemples :
  - La classe V9 contient les verbes qui ont la construction de base *N0 V que P à N2*  
ex. *dire*
  - La classe V31H contient les verbes intransitifs de construction de base *N0 V* avec N0 obligatoirement humain  
ex. *abandonner*

# Lexique-grammaire - 3

- Chaque classe de prédicats est encodée dans une table
  - Ligne : entrée lexicale
  - Colonne : propriété
  - Cellule : valeur booléenne (+ ou -) ou valeur lexicale
- Propriétés syntaxiques étudiées :
  - Nature des arguments (ex. GN, infinitive, complétive)
  - Transformations (ex. passivation, pronominalisation)
  - etc.

# Lexique-grammaire - 4

- Principe de séparation des entrées : une entrée lexicale par sens (sur critères formels)

ex. *acheter*

ACHETER-1 : N0hum acheter N1hum

ex. *Max achète son adversaire (pour gagner le match)*

ACHETER-2 : N0hum acheter (N1hum + N1-hum)

ex. *Max achète (un esclave + une carafe)*

# Exemple de table - V4

	N0 =: Nhum	N0 =: Nnr	N0 =: le fait Qu P	N0 =: V1-inf W	/	Ppv	%	<ENT>	/	V'concret'	N0 V	-a =: ant	-a =: able	-a =: eux	-a =: (E+at)eur	N1 =: Nhum	N1 =: N-hum	N1 =: le fait Qu P	N1 se V de ce Qu P	N1 se V auprès de N3 de ce Qu P	N1 est Vpp de ce Qu P	[passif par]	[passif de]	N0 V N1 contre Nhum
+	+	+	+	+	/	<E>	%	abasourdir	/	-	+	+	-	-	-	+	-	-	-	-	+	+	-	-
+	+	+	+	+	/	<E>	%	abattre	/	+	+	-	-	-	-	+	+	-	+	-	+	+	-	-
+	+	+	+	+	/	<E>	%	abêtir	/	-	+	+	-	-	-	+	-	-	+	-	+	+	-	-
+	+	+	+	+	/	<E>	%	abîmer	/	+	+	-	-	-	-	+	+	-	-	-	-	+	-	-
+	+	+	+	+	/	<E>	%	abuser	/	-	+	-	-	-	-	+	-	-	-	-	-	+	-	-
+	+	+	+	+	/	<E>	%	accaparer	/	+	+	+	-	-	-	+	+	-	-	-	-	+	-	-
+	+	+	+	+	/	<E>	%	accomplir	/	-	+	+	-	-	-	+	-	-	+	-	-	+	-	-
~	~	~	~	/	<E>	%	accrasser	/	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
+	+	+	+	/	<E>	%	accrocher	/	+	+	+	-	-	-	-	+	+	-	-	-	-	+	-	-

# Exemple de table - V31H

N0 être V-n	/	Ppv	%	<ENT>	/	N0 être V-ant	N0 est Vpp	N0pc lui V	N0 V de N0pc	Nhum V sur ce point	N0 V vers N	il V N0 W	N0 =: idée, N1 =: esprit	N0hum V Loc Nq	N0 =: N-hum
-	/	<E>	%	abandonner	/	-	-	-	-	-	-	-	-	-	-
~	/	s'	%	absenter	/	~	~	~	~	~	~	~	~	~	~
-	/	s'	%	abstenir	/	-	-	-	-	+	-	+	-	-	-
-	/	<E>	%	abuser	/	-	-	-	-	+	-	-	-	-	-
~	/	<E>	%	accoucher	/	~	~	~	~	~	~	~	~	~	~
-	/	<E>	%	acquiescer	/	-	-	-	+	+	-	-	-	-	-
~	/	s'	%	adoniser	/	~	~	~	~	~	~	~	~	~	~
-	/	<E>	%	adouber	/	-	-	-	-	-	-	-	-	-	-
-	/	<E>	%	agioter	/	-	-	-	-	-	-	+	-	-	-



# Exemple de table - V33

	N0 =: Nhum	N0 =: N-hum	N0 =: Nnr	N0 être V-n	N0 =: Npl obl	/	Ppv	%	<ENT>	/	N0 V	N0 être V-ant	N0 est Vpp	N0 V de N0pc	N1 =: Nhum	N0 lui V Prép N1pc	N1 =: N-hum	N1 =: le fait Qu P	Ppv =: lui	Ppv =: y	N1 être V-n	N1 =: Npl obl	Nhum V sur ce point	il V N0 W	N0 =: idée, N1 =: espi
~	~	~	~	~	~	/	s'	%	absinther	/	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
+	-	-	-	-	-	/	<E>	%	accéder	/	-	-	-	-	-	-	+	-	-	+	-	-	+	+	-
-	-	-	-	-	-	/	<E>	%	accrocher	/	+	-	+	-	+	-	+	-	-	+	-	-	+	+	-
~	~	~	~	~	~	/	s'	%	acliquer	/	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
+	-	-	-	-	-	/	<E>	%	adhérer	/	-	-	-	-	-	-	+	-	-	+	-	-	-	+	-
-	+	+	-	-	-	/	s'	%	adresser	/	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-
+	-	-	-	-	-	/	s'	%	adresser	/	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-
+	-	-	-	-	-	/	s'	%	adresser	/	-	-	-	-	+	-	-	-	-	+	-	-	+	-	-
+	-	-	-	-	-	/	s'	%	affronter	/	-	-	+	-	+	-	+	+	-	+	-	-	+	-	-
~	~	~	~	~	~	/	les	%	aligner	/	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
+	+	+	-	-	-	/	<E>	%	aller	/	-	-	-	-	+	-	+	-	+	-	-	-	-	+	+
~	~	~	~	~	~	/	les	%	allonger	/	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
+	+	-	-	-	-	/	<E>	%	apparaître	/	-	-	-	-	+	-	-	-	+	-	-	-	-	+	+
+	+	+	-	-	-	/	<E>	%	appartenir	/	-	-	-	-	+	-	+	-	+	-	-	-	-	+	+
~	~	~	~	~	~	/	<E>	%	appartenir	/	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
+	+	-	-	-	-	/	<E>	%	appartenir	/	-	-	-	-	+	+	-	+	+	-	+	-	+	+	-
+	-	-	-	-	-	/	en	%	appeler	/	-	-	-	-	+	-	-	+	-	-	-	-	+	+	-
-	+	+	-	-	-	/	s'	%	appliquer	/	+	-	-	-	+	-	+	+	-	+	-	-	-	-	-
+	+	-	-	-	-	/	<E>	%	assister	/	-	-	-	-	-	-	+	+	-	+	-	-	-	-	-

# Problème

- Il y a des propriétés manquantes !
  - des propriétés définitionnelles des classes
  - des propriétés qui peuvent se déduire d'autres propriétés
  - autres
- Combler ces manques en construisant une table des tables (en cours de construction à l'IGM)

## Exemples - 1

- La propriété définitionnelle de la table V4 est :

*(N-hum+que P+V1inf)0 V N1hum*  
*=: que Luc soit blessé abat Lea*

La propriété N0 V N1 => + est absente !

- La propriété définitionnelle de V33 est :

*N0 V à N1 => +*  
*=: Marie assiste à la rencontre*

Elle est absente !

## Exemples - 2

- La propriété définitionnelle de la table V31H est :

N0 V avec N0 humain obligatoire  
ex. *Luc abandonne*

- Les propriétés

N0 =: Nhum => +

N0 =: N-hum => -

sont absentes !

# La table des tables

- La table des tables permet d'indiquer à chaque table les propriétés constantes de manière explicite
- Elle comprend :
  - Sur les lignes : les noms des différentes tables
  - Sur les colonnes : l'ensemble des propriétés
- La valeur des cellules est :
  - “+” si la propriété est toujours VRAI pour la table donnée
  - “-” si la propriété est toujours FAUSSE pour la table donnée
  - “o” si la propriété est variable et codée dans la table
  - Valeur lexicale

# Echantillon de la table des tables

table	N0 =: Nhum	N0 =: N-hum	N0 =: Nnr	<ENT>	Ppv	N0 V	N0 V N1	N0 V à N1	N1 =: Nhum	N1 =: N-hum
V_4	0	+	+	0	0	0	+	-	0	0
V_31R	0	0	-	0	0	+	-	-	-	-
V_31H	+	-	-	0	0	+	-	-	-	-
V_33	0	0	0	0	0	0	-	+	0	0

# Méthodes existantes de reformatage

- Travaux existants : Hathout et al. (1998) et Gardent et al. (2006)

- Méthodologie utilisée :

pour chaque table, écriture d'une configuration de reformatage avec explicitation des propriétés vraies constantes

- Limites :
  - un travail de configuration pour chaque table !
  - réutilisation difficile de l'explicitation des propriétés manquantes

# Notre méthode

- Utilisation de la table des tables
- Une configuration générale à l'aide d'un petit langage :
  - Définition d'objets linguistiques de base
  - Définition d'opérations sur ces objets
  - Définition du formatage
- A chaque propriété sélectionnée, on associe des objets linguistiques à des opérations
- Limite ? Complexité potentielle (non mesurée)



# Algorithme

lecture des objets

lecture des opérations

pour chq table t de la table des tables

  pour chq entrée de la table t

    copie des objets;

    pour chq propriété sélectionnée (+)

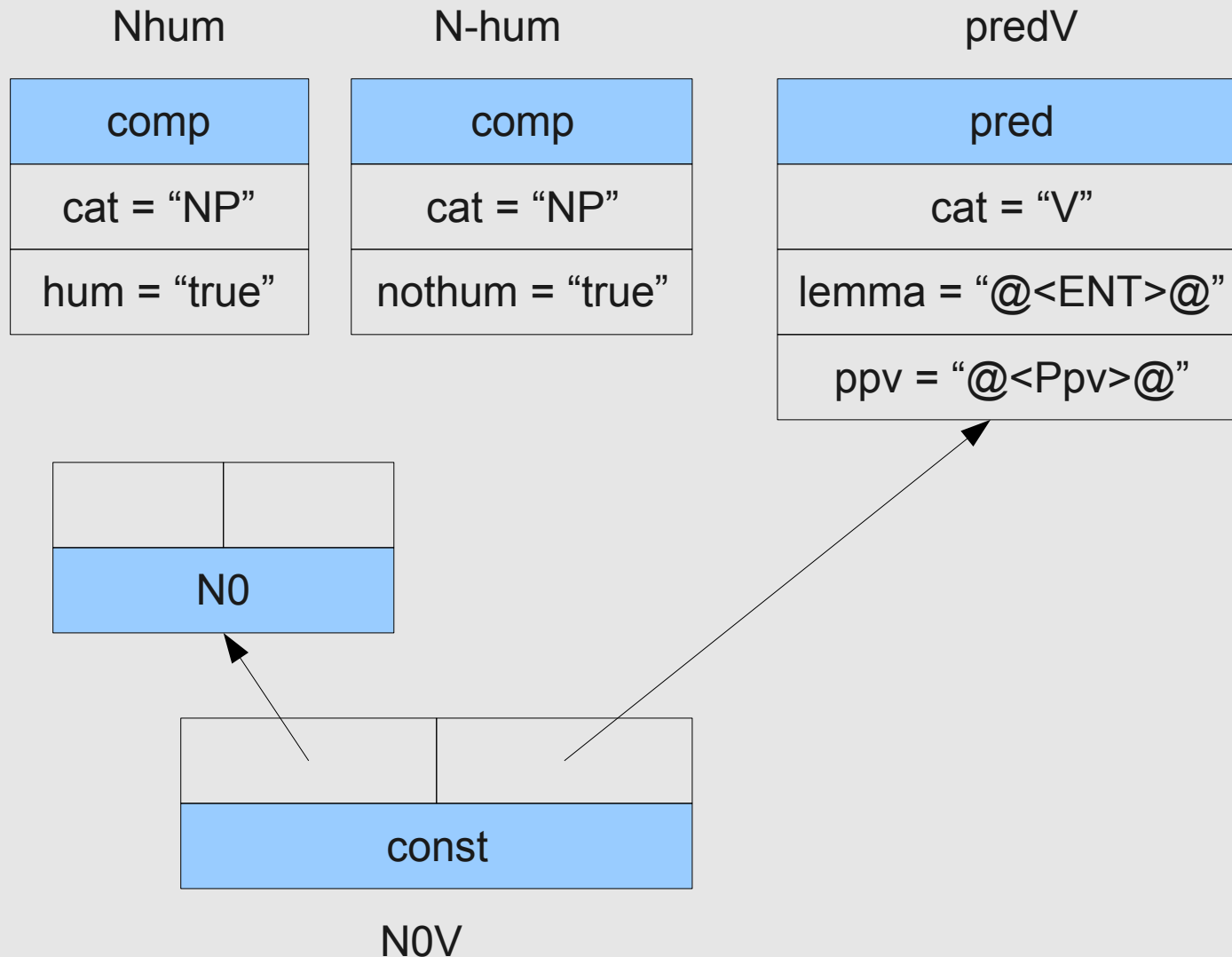
      faire les opérations associées à la propriété  
      écrire le résultat de l'entrée

# Objets linguistiques de base

- Ils sont sous la forme de listes et de structures de traits et ont un type (ex. comp) et un nom (ex. Nhum)
- Ces objets peuvent être paramétrés par les propriétés de la table des tables
- Exemple :

```
define comp Nhum [cat="NP",hum="true"];  
define comp N-hum [nothum="true",cat="NP"];  
define NO N0 ();  
define pred predV [cat="V",lemma="@<ENT>@"];
```

# Exemple : définition des objets de base



# Définition des opérations - 1

- Opérations possibles :
  - Créer un objet dans le lexique
  - Ajouter un trait ou un objet dans une liste
  - Ajouter un trait ou un objet dans une structure de traits
- Propriétés générales des opérations :
  - Non destructives
  - Résultat indépendant de l'ordre d'application des opérations

# Ajout d'un trait ou objet dans une liste

- Algorithme
  - On teste si le trait ou l'objet est déjà présent
  - Si non présent, ajout en fin de liste
- Exemple :  
liste = (a,d,c,b)  
add(a,liste)  
add(f,liste)
- Résultat :  
  
(a,d,c,b,e)

# Ajout d'un trait dans une structure de trait

- Algorithme :
  - On teste si l'attribut n'est pas déjà défini
  - Si non défini, on ajoute le trait
  - Si défini, on fait une disjonction
- Exemple :  
fs = [cat="NP"]  
add(subcat="hum",fs)  
add(subcat="nothum",fs)
- Résultat :  
[cat="NP",subcat="hum | nothum"]

# Ajout d'un objet dans une structure de traits

- Algorithme :
  - Teste si le type de l'objet existe déjà comme attribut
  - S'il existe, on teste si l'objet associé est différent de l'objet à ajouter ; dans ce cas, ERREUR
  - S'il n'existe pas, on ajoute l'objet dans la structure de traits avec comme attribut son type

- Exemple

```
define dist dist0 (N-hum,Nhum) ;  
define const N0 [pos="0"] ;  
add dist0 in N0 ;
```

```
[pos="0",dist=(N-hum,Nhum)]
```

# Définition des opérations sur les objets - exemple

- Si la propriété  $N0 =: Nhum$  est sélectionnée
  - Ajouter Nhum à N0
- Dans le petit langage,

```
prop @N0 =: Nhum@{  
  add Nhum in N0;  
}
```



# Définition des opérations sur les objets - exemple

- Si la propriété *NO V* est sélectionnée
  - Créer la construction *NOV* dans l'entrée
- Dans le petit langage,

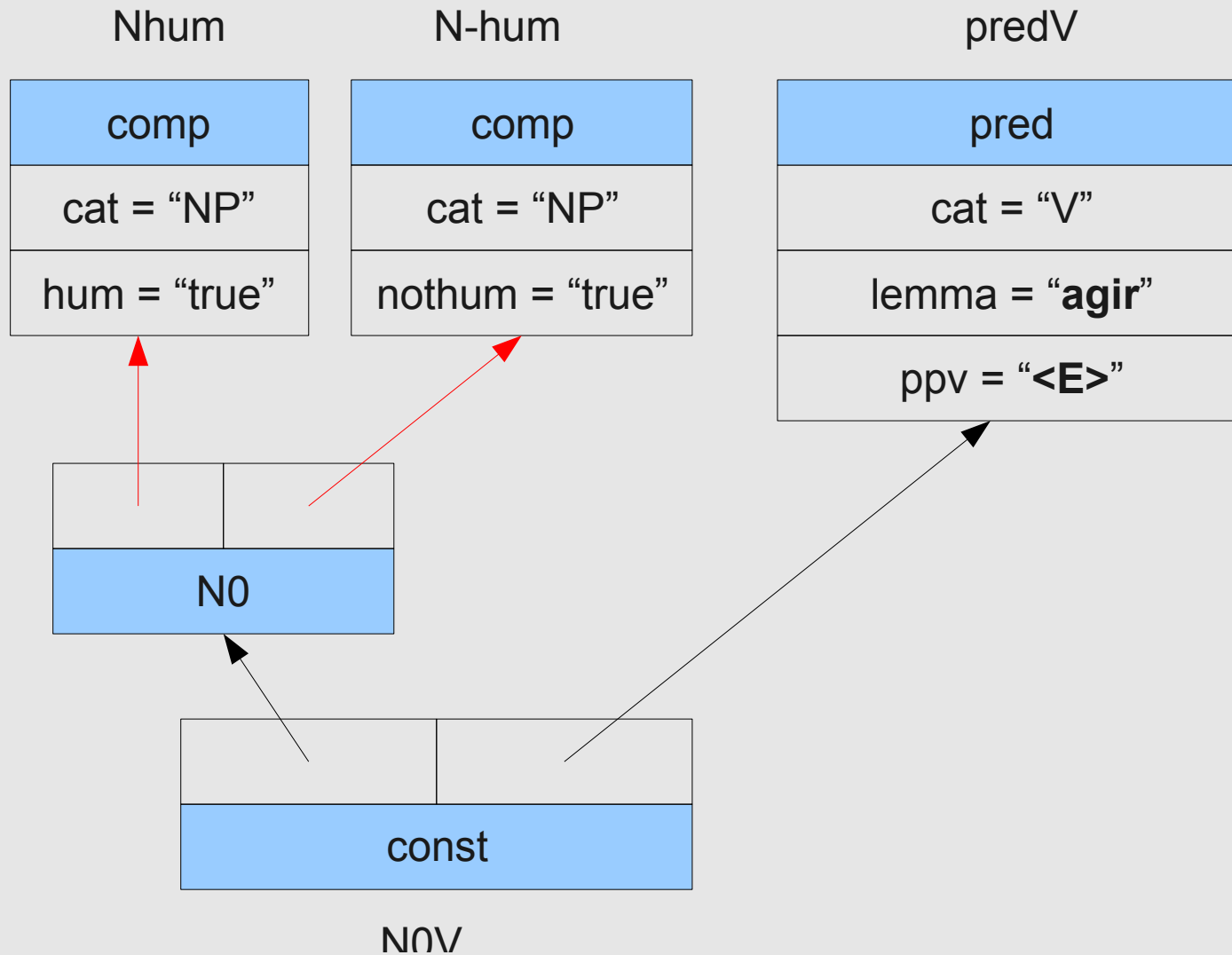
```
define const NOV (NO, predV) ;
```

```
prop @NO V@{  
    create NOV;  
}
```

# Echantillon de la table des tables

table										
V_31R	o	o	.	o	o	+	.	.	.	.
V_31H	+	.	.	o	o	+	.	.	.	.
	N0 =: Nhum	N0 =: N-hum	N0 =: Nnr	<ENT>	Ppv	N0 V	N0 V N1	N0 V à N1	N1 =: Nhum	N1 =: N-hum

# Exemple : entrée agir de V\_31R



# Quelques caractéristiques du langage

- Un objet linguistique peut être inclus dans un autre par simple utilisation de son nom :

```
define comp Nhum [cat="NP",hum="true"];  
define comp N-hum [cat="NP",nothum="true"];  
define dist DIST (Nhum,N-hum);
```

- Une propriété sélectionnée peut activer plusieurs opérations :

```
prop @N0 =: Nhum@{  
    add N0 in args;  
    add Nhum in N0.dist;  
}
```

# Quelques caractéristiques du langage

- Héritage des structures de traits

```
define np NP [cat="NP"] ;  
define comp Nhum NP [hum="true"] ;
```

- Définition et appel de fonctions :

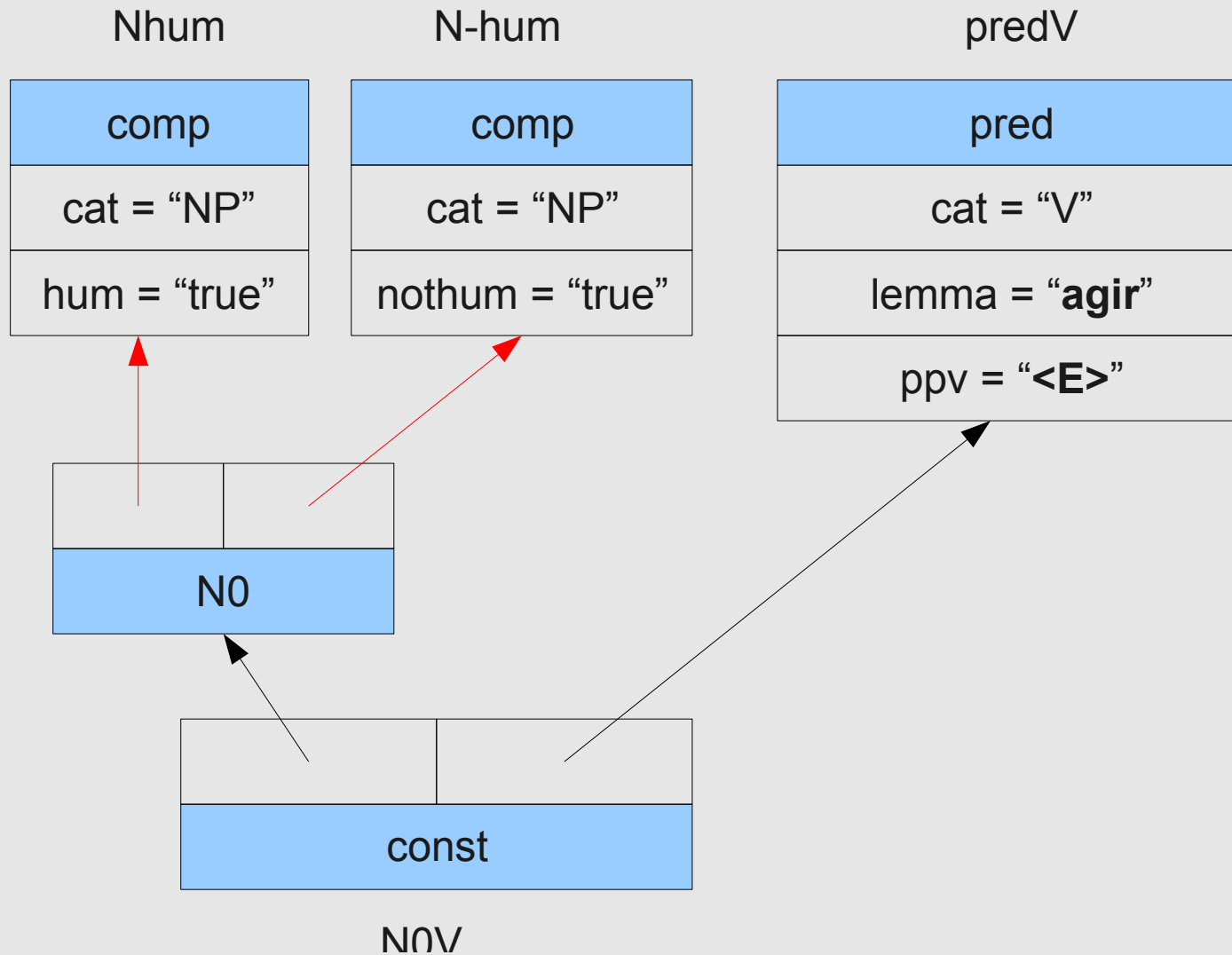
```
prop addNhumInN0 {  
    add Nhum in N0 ;  
}
```

```
prop @N0 =: Nhum@ {  
    addNhumInN0 ( ) ;  
}
```

# Définition du reformatage XML

- Les objets sont des éléments et les traits sont des éléments terminaux particuliers
- A chaque type d'objet, on peut associer
  - un nom d'élément XML
  - différents attributs
- Par défaut, les objets ne sont pas des éléments XML, juste des noeuds intermédiaires

# Exemple : entrée agir de V\_31R



# Exemple

- Dans le petit langage,

```
element pred predicate{}  
element N0 N0{}  
element comp component{cat; }
```

- Signification:

- Les objets du type *pred* sont des éléments nommés *predicate*
- Les objets du type *N0* sont des éléments nommés *N0*
- Les objets du type *com* sont des éléments nommés *component* ; ils ont un attribut *cat* correspondant au trait *cat*



# Résultat du reformatage pour l'entrée “agir”

```
<entry id="V_31R_6">  
  <N0>  
    <component cat="NP">  
      <nothum value="true" />  
    </component>  
    <component cat="NP">  
      <hum value="true" />  
    </component>  
  </N0>  
  <predicate>  
    <lemma value="agir" />  
    <cat value="V" />  
  </predicate>  
</entry>
```

# Conclusion

- Outil utile pour avoir un lexique syntaxique dans différents formats à partir des tables du lexique-grammaire
  - Pas seulement XML : à développer un système de plugin pour le formatage
  - Séparer plus clairement représentation des objets et formatage
- Mais non encore testé à grande échelle pour un lexique-syntaxique pour un analyseur syntaxique en profondeur
  - Table des tables à compléter
  - Spécifier le format et informations nécessaires pour l'analyseur