Grammatical development with XMG : the case of French

Benoit Crabbé

Université Paris 7 and INRIA (Alpage)
SFB 441
SfS
EKU **Tuebingen**
March. 7 2008

Thanks to : D. Duchier (LIFO), C. Gardent (LORIA), Joseph Le Roux (LORIA) and Yannick Parmentier (Tübingen) and too many others. . .

Outline of the talk

- Implementation of large scale computational grammar for French
- Linguistically motivated grammar
- We focus on the implementation of a large scale TAG to be augmented with semantics
- We show how the XMG language can be used to implement and ease the implementation of a real scale grammar of this kind.
- The grammar implemented is a **competence** grammar. It implies that it strictly distinguish between grammatical vs non grammatical sentences. I do not adress the question of parsing real text.

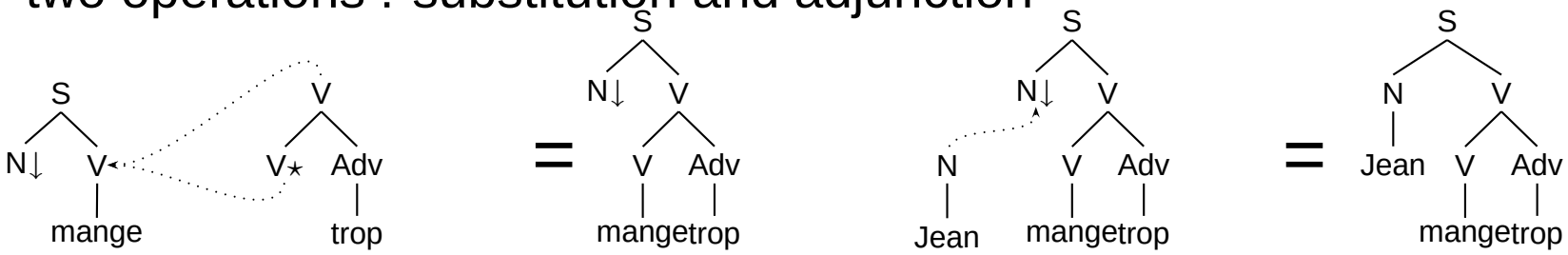
Plan

- **Introduction**
- Requirements and motivations
  - Structure sharing and alternations
- The subset of the language used:
  - A control language and a tree description language
- Methodology
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- Comparisons
  - Métarules $\sim$ Candito and Xia Metagrammars
- Validation :
  - Actual implementation of the grammar and evaluation
- Conclusion

Our specific case : TAG and tree-based formalisms

- For the purpose of natural language parsing, TAG is used in its lexicalised version (LTAG)

- LTAG = All units are lexicalised **elementary** trees. You combine them with two operations : substitution and adjunction



- **Formal result:** (Joshi et Schabès 97; Joshi 2005) prove that LTAG lexicalises strongly a context free grammar. The key of the proof is adjunction.

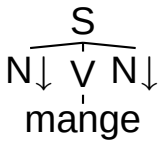
LTAG as a low level formalism

- Formally, an LTAG grammar is a low level grammar for which we have interesting formal properties (lexicalisation) and for which we have efficient parsing algorithms (derived from those used for CFGS)

- However in practice it is insufficient for the purpose of large scale grammatical implementation
  - A raw TAG is made of a very large number of trees reduplicating ever and ever the same blocks of information
  - Lack of expressivity:
    - One cannot express generalisations
    - Raises problems of descriptive redundancy and maintenance of the grammar

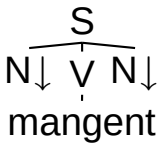
Some trees describing *manger*'s context

(a)



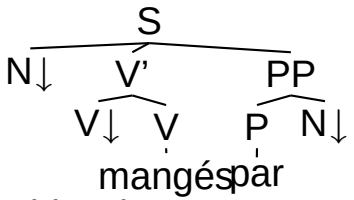
Jean mange des biscuits
John eats the cookies

(b)



Les enfants mangent des biscuits
The children eat the cookies

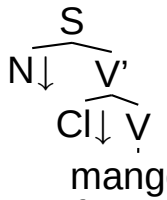
(c)



Les biscuits sont mangés par les enfants
The cookies are eaten by the children

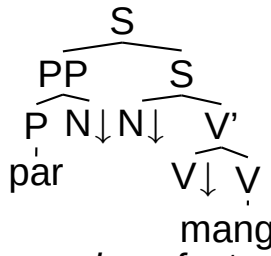
- (a) is a canonical context
- (b) is a plural context
- (c) is a passivised context
- (d) is a clitic argument context
- (e) is a passivised context with wh extraction

(d)



Les enfants les ont mangés
The children have eaten them

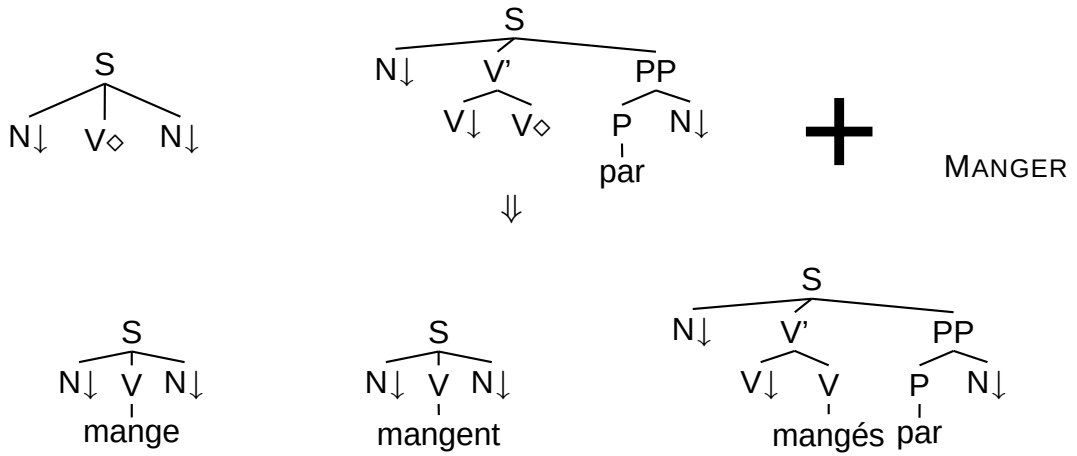
(e)



Par quels enfants les biscuits sont-ils mangés ?
By which children do the cookies have been eaten ?

Implementation : tree schematas and templates

- In practice, implementations (XTAG) split the elementary units between templates and the lexicon
- This first step of factorisation allows to handle morphological variants outside of the grammar (by means of a tokeniser, part of speech tagger)
- Elementary trees are built dynamically (on the fly) by the parser at parse time.
  - The lexicon is made of lemmas, each of them associated to (at least) **a tree family** representing its possible alternative contexts



Metagrammar is about describing templates

- In current implementations, a Tree Adjoining Grammar is a set of templates organised in families
- We get rid of morphological issues at preprocessing
- However in a realistic grammar, the number of templates remain quite high. (thousands, millions, or billions in MGCOMP)
- For maintenance reasons and to ease the design of the grammar we need an additional language that allows to express generalisations among these **templates** (metagrammar)

Plan

- Introduction
- **Requirements and motivations**
  - Structure sharing and alternations
- The subset of the language used:
  - A control language and a tree description language
- Methodology
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- Comparisons
  - Métarules $\sim$ Candito and Xia Metagrammars
- Validation :
  - Actual implementation of the grammar and evaluation
- Conclusion

High Level vs Low Level languages (ANLT)

- from John Caroll's Phd (**1993**), a reference GPSG implementation:
  - A **low level language :**
    - Well defined (syntax and semantics) : interface with parsing algorithms
    - e.g. CFG + atomic feature structures
    - . . . coded on integers
  - A **high level language** (dubbed metagrammar as in GKPS) :
    - Principles : Head Feature propagation, valency, slash (unbounded dependencies)
    - + metarules (allowing to express alternations such as active / passive)
  - Why two languages ?
    - The low level language is interfaced with a parser and does not change. It lacks of expressivity for the linguist however.
    - The high level language is more *"sloppy"*. It provides more expressive power to the linguist.
    - The high level language is more flexible : it may change according to the linguist needs.

Structure sharing and alternations (PATR II)

- PATR II (**Shieber 85**)
  - How does high level languages look like ? The simplest high level language
  - Computer language without any theoretical goal.
  - Data structure = feature structures
  - Allows to express :
    - **Structure sharing** (parametrized macros) : allow to implement inheritance hierarchies

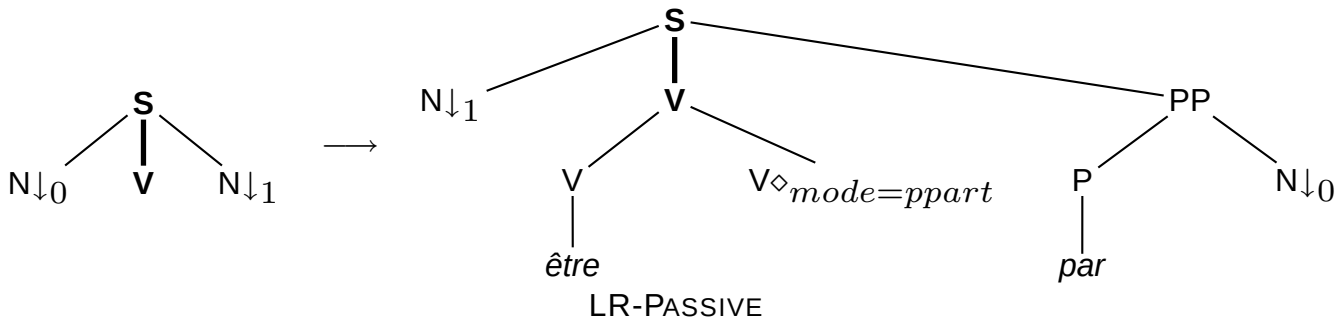
```
VERB : <cat> = v
VERB-3RD-SING :
  @VERB
  <agr num> = singular
  <agr person> = 3
```

    - **Alternations** (lexical rule: yields a new `out` passive entry from the `in` active entry)

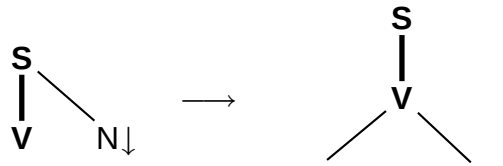
```
Passive :
  <out cat> = <in cat>
  <out form > = past-prt
  <out subj> = <in obj>
  <out obj> = <in subj>
```

Example TAG : high level apparatus (Becker 93)

- Structure sharing : **canonical trees** are organised in an inheritance hierarchy (like PATR macros)
- **Metarules** allow to express alternations $\approx$ transformations compiled offline prior to parsing restricted to a local domain) yielding additional elementary trees. (procedural and shown to be undecidable)



LR-PASSIVE



LR-CLITIC(OBJ)

Structure sharing and alternations

- In traditional post-generative (unification) formalisms we find the need to express two axis for representing the information
  - An axis representing structure sharing
    - Example : a transitive verb as an intransitive verb share the common information that they are verbs
  - An axis representing alternations :
    - Example : a passive verb is an alternate realisation of a transitive active verb
- Classical formalisation
  - Structure sharing is formalised usually as an inheritance hierarchy
  - Alternations are usually formalised with lexical rules
- Main Motivation (XMG) : be declarative, avoid procedural devices
- Second motivation (XMG) : keep those two axes as primitives of the language (reuse previous grammatical descriptions)

Plan

- Introduction
- Requirements and motivations
  - **Structure sharing and alternations**
- The subset of the language used in this talk:
  - A control language and a tree description language
- Methodology
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- Comparisons
  - Métarules $\sim$ Candito and Xia Metagrammars
- Validation :
  - Actual implementation of the grammar and evaluation
- Conclusion

Plan

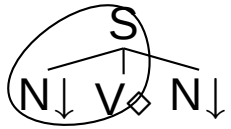
- Introduction
- Requirements and motivations
  - Structure sharing and alternations
- **The subset of the language used**
  - A control language and a tree description language
- Methodology
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- Comparisons
  - Métarules $\sim$ Candito and Xia Metagrammars
- Validation :
  - Actual implementation of the grammar and evaluation
- Conclusion

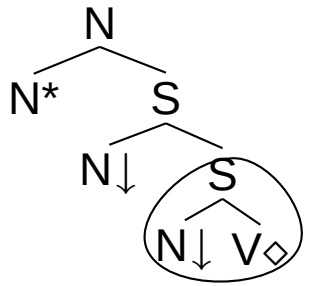
The XMG language

- We use a grammatical description language
  - That allows to represent
    - Structure sharing
    - Alternations
- Formally :
  - Two languages are combined:
  - A **control language** that is interpreted as a logic program
  - A **tree description language** that is cast as a constraint satisfaction problem

Structure sharing

- Structure sharing:



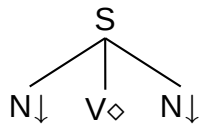


Jean mange des biscuits — *Les biscuits que Jean mange*
John eats cookies — *The cookies that John eats*

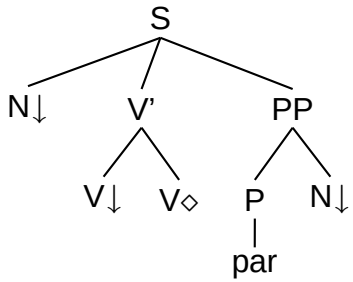
- We wish to indentify and to reuse tree fragments shared by many trees in the grammar (like the canonical subject)

Alternations

- Alternatives :



tree representing the active

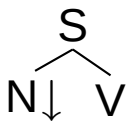


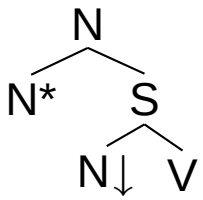
tree representing the (by) passive

- Alternations have a specific status :
  - They contribute to describe tree sets. Methodologically those trees are related to each other (≈ generally speaking they share the same semantics)
  - A TAG family is a set of trees describing alternative realisations of the same subcategorisation frame.

The control language

- Allows to **name** grammatical descriptions

(1)a. CanonicalSubject → 

b. RelativisedSubject → 

c. ActiveForm → 

- A named description (or class) can be reused elsewhere (in a similar but not equivalent fashion as a macro)

Combining descriptions

- **Disjunction (choice) of descriptions**

  (2) Subject $\rightarrow$ CanonicalSubject $\vee$ RelativisedSubject

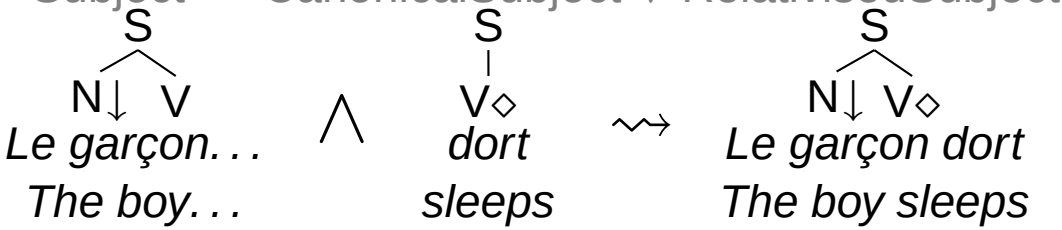
  A subject is either a canonical subject or a relativised subject. Disjunction is a **choice** (nondeterministic interpretation)

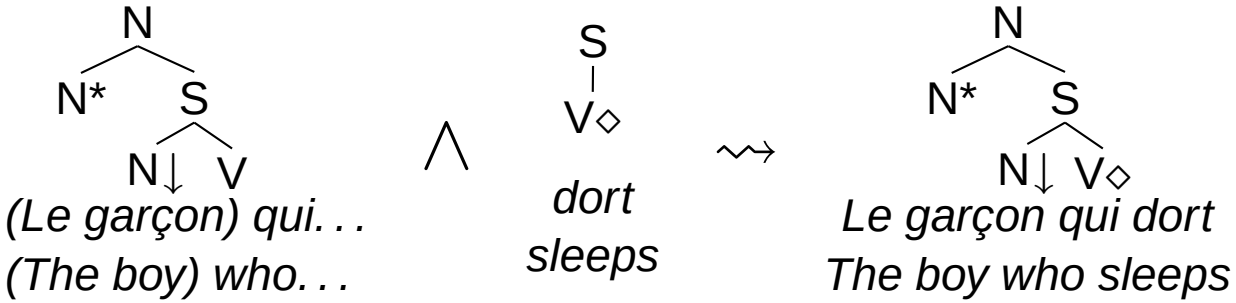
- **Conjunction of descriptions**

  (3) IntransitiveVerb $\rightarrow$ Subject $\wedge$ ActiveForm

  A conjunction of descriptions is interpreted as a syntactic conjunction of two tree descriptions where the name of the nodes are renamed.

Example of interpretation

- Valuation of the class *IntransitiveVerb* :
- (Remember) :
  - IntransitiveVerb $\longrightarrow$ Subject $\wedge$ ActiveForm
  - Subject $\longrightarrow$ CanonicalSubject $\vee$ RelativisedSubject
- 



Tree description language

- Here we answer two questions : what are these fragments ? How do they get combined together ?
- = “classical” language of tree descriptions
  - Specificity (vs Candito 99, Xia 01) : when combining two descriptions ($\wedge$) nodes are renamed

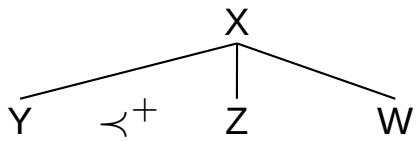
    $\rightsquigarrow$ allows to reuse several times the same class in order to generate a single tree
  - This classical language is further augmented with additional properties and constraints that are aimed at ensuring the tree well formedness

The basic language

- It is a logic that allows to talk about trees. The basioc languages includes relations such as reflexive transitive dominance, immediate dominance, precedence, adjacency (binary relations) and labelling (unary relation)
  - The labelling relation involves labelling with complex categories (feature structures)
- Notation :

(D_0) $\quad y \prec^+ z \wedge z \prec w \wedge x \triangleleft^* y \wedge x \triangleleft y \wedge x \triangleleft w \wedge x : X \wedge y : Y \wedge z : Z \wedge w : W$

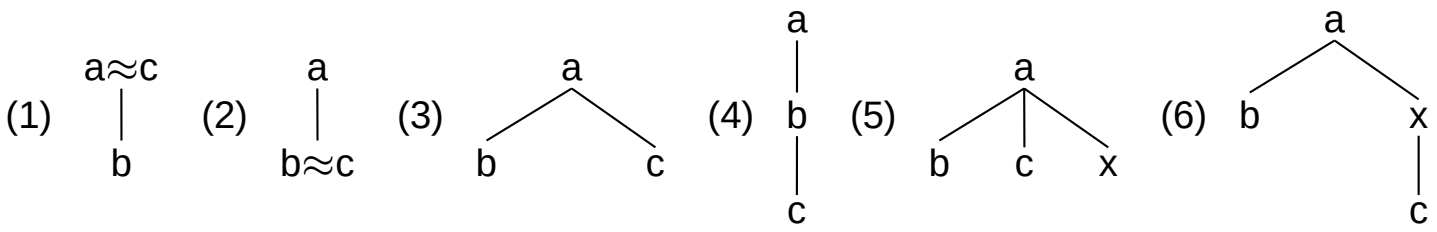
is depicted as : (D_0)



- A formula in this language is interpreted as finite minimal model

Minimal model

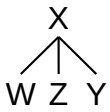
- Given a formula, one can look for the **class** of models (= being finite linear ordered trees) that satisfy the formula.
- This set is generally infinite (or null if the formula is a contradiction)
- A minimal model :
  - Minimises the number of nodes
  - Minimises linear dominance
- Example : $a \lhd b \wedge a \lhd^* c$

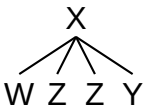


Extension of the basic language to handle naming problems

- Recall that a class in the metagrammar defines its own namespace
- When combining two descriptions we rename everything
- Example : Two descriptions whose names have been anonymised:

- This yields (with root unicity and category unification constraints) :

(a) 

(b) 

- We do not want to keep (b)

Additional constraints

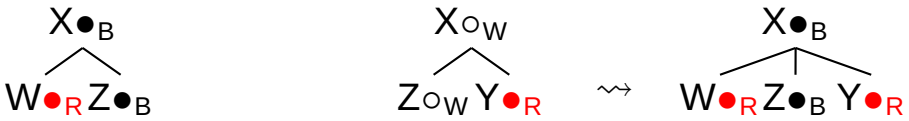
- For both formal and practical reasons the basic language turns out to be insufficient.
- We allow to parametrise it with additional constraints, that is:
  - A set of additional unary properties associated to the nodes
  - We take advantage of these additional properties to define new constraints of well-formedness in order to further enforce tree model admissibility.
  - Examples :
    - Colouring constraints, unicity of the extracted arguments, clitic ordering constraints, wh islands.

Coloring constraints (goal)

- Comes from polarity system (e.g. Interaction Grammars)
- Introduction of a combination schema :
  - Each node of the description is associated to a property, a color (white, black, red)
  - Constraint : Each resulting node in a model is colored either in black or in red.
  - When two nodes are merged, the colors are merged as follows:

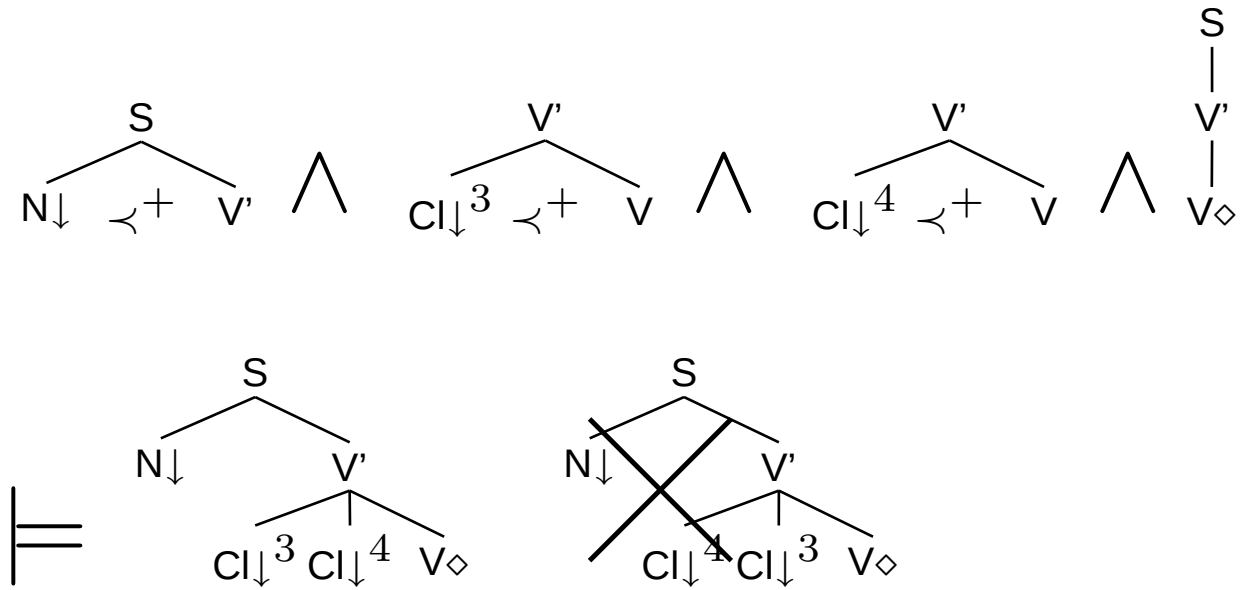
	$\bullet_B$	$\bullet_R$	$\circ_W$
$\bullet_B$	$\bot$	$\bot$	$\bullet_B$
$\bullet_R$	$\bot$	$\bot$	$\bot$
$\circ_W$	$\bullet_B$	$\bot$	$\circ_W$

- The red represents total saturation, black partial saturation (such a node may be optionally combined with another one) and white non saturation
- Example :



Clitic ordering constraint

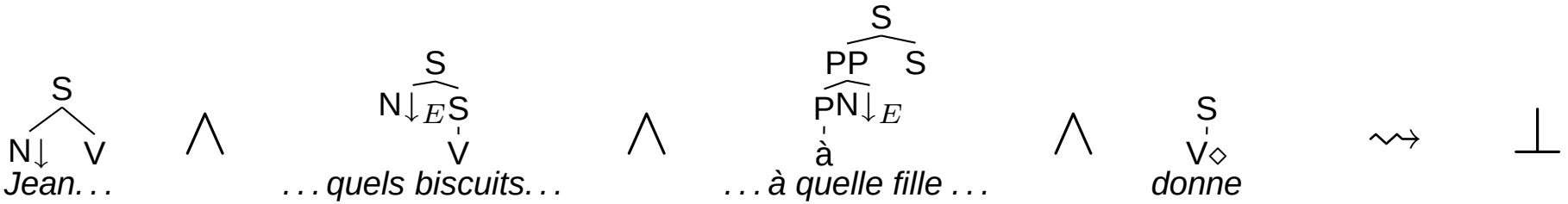
- Clitic ranks = unary properties
- Constraints = linear order defined over the rank property of sibling nodes



Unicity of the extracted argument

- Multiple extractions are so uncommmon in French that it is better to rule them out of the grammar

  ** A quelle fille Quels biscuits donne Jean ?*
  ** To which girl which cookies does John give ?*
- We use a unicity principle :
  - Property attached to the node: E
  - Constraint : a resulting model cannot contain more than a single node marked with that property.



Classifying the constraints

- These constraints have a direct inspiration from LFG/GPSG (Kaplan, Gazdar,Pullum)
- Classification (inspiration from G.K.Pullum) :
  - Formal constraints on data structures = constraints expressed in the basic language to ensure “treeness”
  - Operational (naming) constraint: coloring
  - Universal constraints ($\approx$ principles) : ex. completeness/unicity in LFG, (Frank 02) for TAG. . . not expressed in XMG
  - Language specific constraints ($\approx$ parameters) : ex. clitic ordering,extraction unicity etc.
- The implementation is designed to allow the addition of new constraints (in a programatic fashion). Hence XMG
- A similar idea is introduced in XDG (Debussman et. al) who views parsing as a constraint satisfaction problem : here we apply the constraints offline.

Plan

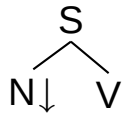
- Introduction
- Requirements and motivations
  - Structure sharing and alternations
- The subset of the language used in this talk:
  - A control language and a tree description language
- **Methodology**
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- Comparisons
  - Métarules $\sim$ Candito and Xia Metagrammars
- Validation :
  - Actual implementation of the grammar and evaluation
- Conclusion

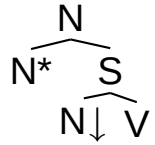
Methodology (introduction)

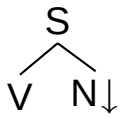
- We show that the XMG language allows to reuse massively the methodology introduced by (Candito 99) and (Xia 01) for describing their grammars.
- We proceed in four steps :
  - Describing and organising primitive tree fragments
  - Describing functional alternatives
  - Describing diathesis alternatives
  - Describing tree families

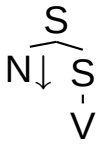
Tree fragments (building blocks)

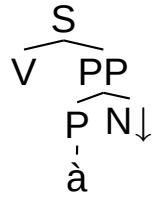
Each fragment is associated to a name that allows to reuse it afterwards

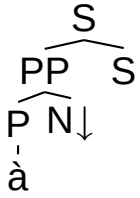
SubjCanon → 

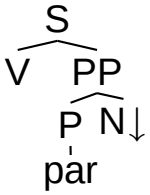
SubjectRel → 

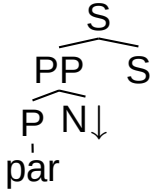
ObjCanon → 

ObjWh → 

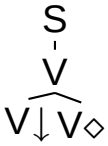
ObjIndCanon → 

ObjIndWh → 

ByObjCanon → 

ByObjWh → 

ActiveForm → 

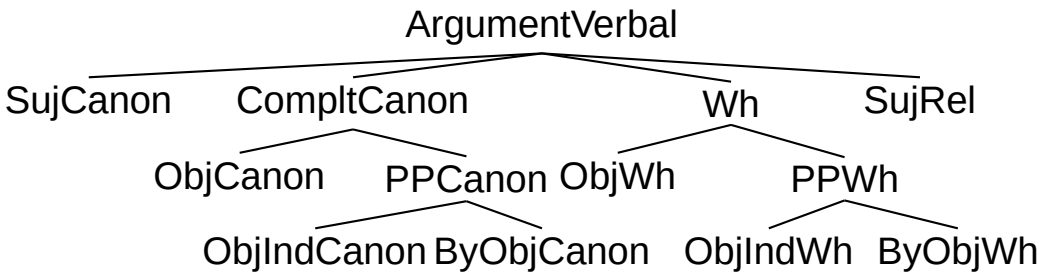
PassiveForm → 

Organising fragments in an inheritance hierarchy

- Example :
  - IndCanonObj $\longrightarrow$ PPCanon $\wedge$ ϕ

    ϕ is the additional information that stands for specialisation

- $\rightsquigarrow$ We say **informally** that *ObjIndCanon* inherits from *PPCanon*



- In an inheritance context we also use an additional device that allows a subclass to access the names declared in the upperclasses (through the import/export device).

Syntactic functions

- Synytactic functions are viewed as abstractions over actual syntactic realisations

  (4)a. Subject $\longrightarrow$ SubjCanon $\vee$ SubjRel
  b. Object $\longrightarrow$ ObjCanon $\vee$ ObjWh
  c. ByObject $\longrightarrow$ ByObjCanon $\vee$ ByObjWh
  d. IndirectObject $\longrightarrow$ ObjIndCanon $\vee$ ObjIndWh

- For instance, *IndirectObject* stands for alternations such as :

  (5)a. Jean parle **à Marie** (Objet indirect canonique)
  b. *John speaks* **to Mary** *(canonical indirect object)*
  c. **A qui** Jean parle-t-il ? (Objet indirect wh)
  d. **To whom** *does John speak ?* *(wh indirect object)*

Diathesis alternations

- Here we deal with alternations such as active/passive, impersonal causative and the like.

  (6) TransitiveAlternation $\longrightarrow$
  (Subject $\wedge$ ActiveForm $\wedge$ Objet)
  $\vee$(Subject $\wedge$ PassiveForm $\wedge$ ByObject)

- It says that the active is realised by the a subject a verb with an active morphology and a direct object while at the passive we have a subject a verb at the passive form and a *By Object*:

  (7) a. Jean **envoie** une lettre
  b. John sends a letter
  c. Une lettre **est envoyée** par Jean
  d. A letter is sent by John
  e. Par quelle personne la lettre **est-elle envoyée** ?
  f. By whom does this letter has been sent ?

TAG **Families**

- Finally we can handle TAG families (sets of trees sharing the same subcat frame)

  (8) DitransitiveFamily $\rightarrow$ TransitiveAlternation $\wedge$ IndirectObject

- A TAG family is an abstraction that models all the possible alternative realisations of a subcategorisation frame:

  (9)a. Jean offre des fleurs à Marie
  b. John offers flowers to Mary
  c. A quelle fille Jean offre-t-il des fleurs ?
  d. To which girl does John offer flowers .
  e. Par quel garçon les fleurs sont-elles offertes à Marie ?
  f. By which boy do the flowers have been offered to Mary ?

Plan

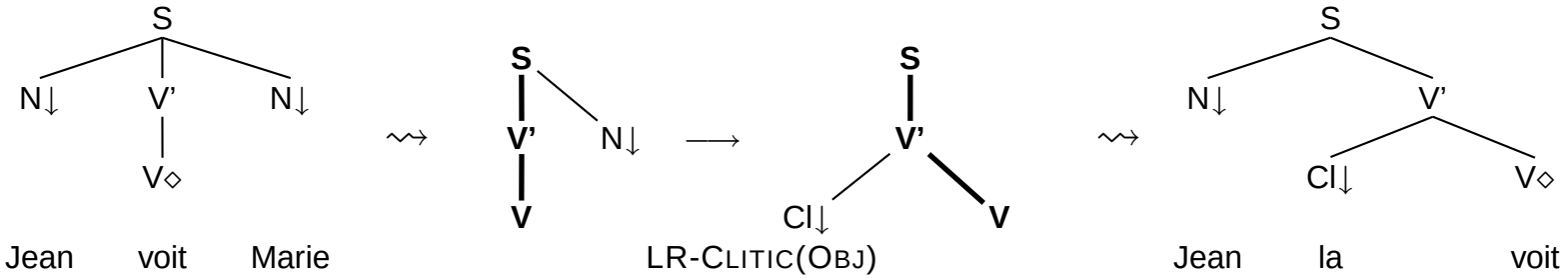
- Introduction
- Requirements and motivations
  - Structure sharing and alternations
- The subset of the language used:
  - A control language and a tree description language
- Methodology
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- **Comparisons**
  - Metarules $\sim$ Candito and Xia Metagrammars
- Validation :
  - Actual implementation of the grammar and evaluation
- Conclusion

Comparisons (metarules)

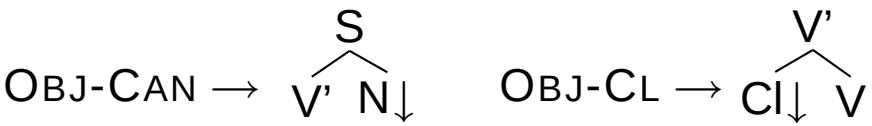
- Metarules (Becker 93) $\approx$ counterpart to lexical rules for TAGS
  - Metagrammar = declarative system, no problem of termination
  - The fragments used in the metagrammar are counterparts to the modified left hand side and right hand side of the metarules
  - The metagrammar provides a way to handle interactions (like clitics) where metarules don't.

So you said you were declarative. . .

- Disjunction allows to manipulate tree sets.
- View lexical rules as operators that yield tree sets out of canonical trees
- Thus :



- Is expressed in XMG as :



Object ⟶ Obj-Can ∨ Obj-Cl

TreeSet ⟶ <The remaining of the tree> ∧ Object

- Without having a canonical tree distinguished

Comparisons (other metagrammars)

- Metagrammars (Candito 99, Xia 99)
  - Here we are fully declarative vs (Candito and Xia)
    - Because we do not distinguish between a canonical and derived trees among a tree set.
    - We have also shown elsewhere that we can perform the syntax semantics interface through linking theory
  - The new thing is mainly the explicit introduction of a choice operator
  - The formal system is not tied to the grammar writing methodology
    - Node naming + colors $\rightsquigarrow$ have reduced node naming problems

Plan

- Introduction
- Requirements and motivations
  - Structure sharing and alternations
- The subset of the language used:
  - A control language and a tree description language
- Methodology
  - Conjunction,disjunctions $\rightsquigarrow$ Structure sharing / alternations
- Comparisons
  - Metarules $\sim$ Candito and Xia Metagrammars
- **Validation :**
  - Actual implementation of the grammar and evaluation
- Conclusion

Validation : a fragment of French grammar

- To test the empirical adequation of the language, I have implemented a significant fragment of French Grammar (TAG, using Candito 99 and Abeillé 02 grammars)
- Overview of the coverage (verbal and adjectival dependants)

Constructions	*Canonical, Clitic, Interrogative, Relative, Cleft*
Syntactic functions	*Subject, Object, Indirect Object, Genitive, Locative, Obliques*
	Sentential Subject, Sentential Objects, indirect interrogatives
Diathesis	*Active, Passive, Impersonal, Middle, Reflexive*
Subcat frames	*46 subcat frames*

- Evaluation with TSNLP (Lehmann 96)
- With a TAG parser called LLP2 (LORIA)
  - Grammatical Items: 76% success
  - Aggramatical Items : Rejects 83 %
  - Average (structural) ambiguity : 1.63

Main causes of failure

- Coordination
- Negation
- Incises
- Comparative
- Causative
- Imperative clitic inversion
- Object control
- Misc :
  - Differences of judgment on the grammaticality of sentences
  - Phonological interactions *trouvé-je*
  - Multi Word expressions
  - Errors in the lexicon

Conclusion

- XMG is a language for grammatical representation that is declarative, it breaks in :
  - A control language (composition, disjunction)
  - A tree description language (augmented with principles)
- The implementation methodology allows to
  - Reuse works in formal and theoretical linguistics
  - Add a semantic layer
- Some nice add-ons :
  - Actual implementation of a semantics (Loria + Tuebingen)
  - extensions to Interaction Grammars + MC-TAGS (Tuebingen)
  - We have syntactic lexicons to use the grammar with
    - Remaining questions : what to put in the grammar (nowadays there is a trend towards “performance” grammars)