A Toolbox for Pregroup Grammars

Une boîte à outils pour développer et utiliser les grammaires de prégroupe

Denis Béchet (Univ. Nantes & LINA)

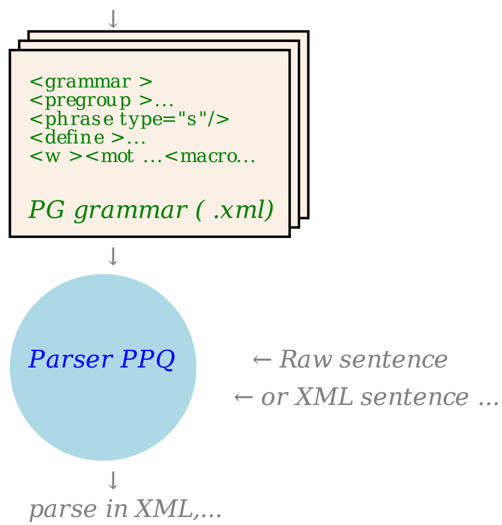
Annie Foret (Univ. Rennes 1 & IRISA)

Denis.Bechet@univ-nantes.fr
http://www.sciences.univ-nantes.fr/info/perso/permanents/bechet

Annie.Foret@irisa.fr, http://www.irisa.fr/prive/foret

Overview

- Grammatical formalism : pregroups
- A pregroup ToolBox : principles and illustrations



  - Majority (partial) composition and parsing
  - Grammar construction
- PPQ demo

Grammatical Formalism

- Pregroup grammars (PG in short) [Lambek 99]:
  - a simplification of Lambek Calculus [1958]
  - used to describe the syntax of natural languages
- Extensions [LATA 2008]:
  we have extended the pregroup calculus with two type constructors that PG are not able to naturally define:
  - * for iteration simple types
  - ? for optional simple types

  and preserve nice properties of PG.

Pregroup : definitions

A *pregroup* $(T, \leq, \cdot, l, r, 1)$
s. t. $(T, \leq, \cdot, 1)$ is a partially ordered monoid [a]
in which l, r are unary operations on T that satisfy:

$$a^l.a \leq 1 \leq a.a^l \quad \text{and} \quad a.a^r \leq 1 \leq a^r.a \qquad (PRE)$$

or equivalently: $a.b \leq c \Leftrightarrow a \leq c.b^l \Leftrightarrow b \leq a^r.c$

Pregroup : definitions

A *pregroup* $(T, \leq, \cdot, l, r, 1)$
s. t. $(T, \leq, \cdot, 1)$ is a partially ordered monoid [a]
in which l, r are unary operations on T that satisfy:

$$a^l.a \leq 1 \leq a.a^l \quad \text{and} \quad a.a^r \leq 1 \leq a^r.a \qquad (PRE)$$

or equivalently: $a.b \leq c \Leftrightarrow a \leq c.b^l \Leftrightarrow b \leq a^r.c$

Some equations follow from the def. $a^{rl} = a = a^{lr}$ [b]

but not, in general: $a^{rr} \neq a \neq a^{ll}$

Iterated adjoints: $\ldots a^{(-2)} = a^{ll}, a^{(-1)} = a^l, a^{(0)} = a, a^{(1)} = a^r, a^{(2)} = a^{rr} \ldots$

---

[a]

Pregroup : definitions

A *pregroup* $(T, \leq, \cdot, l, r, 1)$
s. t. $(T, \leq, \cdot, 1)$ is a partially ordered monoid [a]
in which l, r are unary operations on T that satisfy:

$$a^l.a \leq 1 \leq a.a^l \quad \text{and} \quad a.a^r \leq 1 \leq a^r.a \qquad (PRE)$$

or equivalently: $a.b \leq c \Leftrightarrow a \leq c.b^l \Leftrightarrow b \leq a^r.c$

Some equations follow from the def. $a^{rl} = a = a^{lr}$ [b]

but not, in general: $a^{rr} \neq a \neq a^{ll}$

Iterated adjoints: $\ldots a^{(-2)} = a^{ll}, a^{(-1)} = a^l, a^{(0)} = a, a^{(1)} = a^r, a^{(2)} = a^{rr} \ldots$

---

[a] A partially ordered monoid is a monoid $(M, \cdot, 1)$ with a partial order $\leq$ s. t. $\forall a, b, c$: $a \leq b \Rightarrow c \cdot a \leq c \cdot b$ and $a \cdot c \leq b \cdot c$.

[b] we also have: $(a.b)^r = b^r.a^r \ , \ (a.b)^l = b^l.a^l \ , \ 1^r = 1 = 1^l$

Free pregroup

Let $(P, \leq)$ be an ordered set of atomic types,

Types $T_{(P,\leq)} = \{p_1^{(i_1)} \cdots p_n^{(i_n)} \mid 0 \leq k \leq n, p_k \in P$ and $i_k \in \mathbb{Z}\}$

the empty sequence is denoted by 1.

For X and $Y \in T_{(P,\leq)}$ $\boxed{X \leq Y}$ iff this relation is deductible in the following system

where $\boxed{p, q \in P}$ $n, k \in \mathbb{Z}$ and $X, Y, Z \in T_{(P,\leq)}$:

$$X \leq X \quad (Id) \qquad \frac{X \leq Y \quad Y \leq Z}{X \leq Z}\,(Cut)$$

$$\frac{XY \leq Z}{Xq^{(n)}q^{(n+1)}Y \leq Z}\,(A_L) \qquad \frac{X \leq YZ}{X \leq Yq^{(n+1)}q^{(n)}Z}\,(A_R)$$

$$\frac{Xp^{(k)}Y \leq Z}{Xq^{(k)}Y \leq Z}\,(IND_L) \qquad \frac{X \leq Yq^{(k)}Z}{X \leq Yp^{(k)}Z}\,(IND_R)$$

$q \leq p$ if k is even, and $p \leq q$ if k is odd

Pregroup grammar

Let $(P, \leq)$ be a finite partially ordered set.

- A *pregroup grammar* based on $(P, \leq)$ is a lexicalized[a] grammar $G = (\Sigma, I, s)$ such that
  - $s \in T_{(P,\leq)}$;
  - G assigns a type X to a string $v_1, \ldots, v_n$ of Σ^* iff for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$ such that $X_1 \cdots X_n \leq X$ in the free pregroup $T_{(P,\leq)}$.
- *The language* $\mathcal{L}(G)$ is the set of strings in Σ^* that are assigned s by G.

---

[a] a lexicalized grammar is a triple (Σ, I, s): Σ is a finite alphabet, I assigns a finite set of categories (or types) to each $c \in \Sigma$, s is a category (or type) associated to correct sentences.

Pregroup net

Our example is taken from Lambek, with the atomic types:

π_2 = second person,
p_2 = past participle,
o = object,
q = yes-or-no question,
q' = question $\quad q \leq q'$

This sentence gets type q' ($q' \leq s$):

whom have you seen

$\underline{q'} o^{ll} q^l \quad q p_2^l \pi_2^l \quad \pi_2 \quad p_2\, o^l$

$q' \leq s$

Pregroup net

Our example is taken from Lambek, with the atomic types:

π_2 = second person,
p_2 = past participle,
o = object,
q = yes-or-no question,
q' = question

$q \leq q'$

This sentence gets type q' ($q' \leq s$):

whom have you seen

$\underline{q'} o^{ll} q^l \quad q p_2^l \pi_2^l \quad \pi_2 \quad p_2 \, o^l$

Pregroup net

Our example is taken from Lambek, with the atomic types:

π_2 = second person,
p_2 = past participle,
o = object,
q = yes-or-no question,
q' = question

$$q \leq q'$$

This sentence gets type q' ($q' \leq s$):

$$\begin{array}{cccc} \text{whom} & \text{have} & \text{you} & \text{seen} \\ \underline{q'}\, o^{ll} q^{l} & q\, p_2^{l} \pi_2^{l} & \pi_2 & p_2\, o^{l} \end{array}$$

Pregroup net

Our example is taken from Lambek, with the atomic types:

π_2 = second person,
p_2 = past participle,
o = object,
q = yes-or-no question,
q' = question

$q \leq q'$

This sentence gets type q' ($q' \leq s$):

whom have you seen

$$\underline{q'} o^{ll} q^l \quad q p_2^l \pi_2^l \quad \pi_2 \quad p_2 \, o^l$$

Partial Composition

- [C] **(partial composition)** : for $k \in \mathbb{N}$,

$$\Gamma, X p_1^{(n_1)} \cdots p_k^{(n_k)}, q_k^{(n_k+1)} \cdots q_1^{(n_1+1)} Y, \Delta \xrightarrow{C} \Gamma, XY, \Delta$$

if $p_i \leq q_i$ and n_i is even
or if $q_i \leq p_i$ and n_i is odd ,
for $1 \leq i \leq k$.
Example :

$$\Gamma, \boxed{q' o^{ll} q^l, q p_2^l \pi_2^l}^{[1]}, \Delta \xrightarrow{C} \Gamma, q' o^{ll} p_2^l \pi_2^l, \Delta$$

Majority (Partial) Composition

A partial composition $\xrightarrow{C}$ is a *majority partial composition* ($\xrightarrow{@}$) if the width of the result is not greater than the maximum widths of the arguments

A partial composition that is not a majority composition :

$$\Gamma, \boxed{q' o^{ll} q^{l}, q p_2^{l} \pi_2^{l}}^{[1]}, \Delta \xrightarrow{C} \Gamma, q' o^{ll} p_2^{l} \pi_2^{l}, \Delta$$

A majority composition :

$$\Gamma, \boxed{q' o^{ll} q^{l}, q o^{l} \pi_2^{l}}^{[2]}, \Delta \xrightarrow{@} \Gamma, q' \pi_2^{l}, \Delta$$

Parsing using Majority Composition

Parsing of “whom have you seen ?” ($q' \leq s$)

whom	have	you	seen
$q'o^{ll}q^l$	$qp_2^l\pi_2^l$	π_2	p_2o^l

$$\boxed{\boxed{q'o^{ll}q^l,\ \boxed{qp_2^l\pi_2^l, \pi_2}^{[1]}}^{[1]} , p_2o^l}^{[2]}$$

Parsing algorithm in n^3

1. Types for words

whom	$\mapsto$	$\{q'o^{ll}q^{l}\}$
have	$\mapsto$	$\{qp_2^l\pi_2^l\}$
you	$\mapsto$	$\{\pi_2\}$
seen	$\mapsto$	$\{p_2o^l\}$

2. Add types for words with $\xrightarrow{GCONC^+}$

3. Rec. Calculus of types per segment, with $\xrightarrow{@}$

4. Test wether the sentence has atomic type s or $x \leq s$

Parsing algorithm in n^3

1. Types for words

whom	$\mapsto$	$\{q' o^{ll} q^l\}$
have	$\mapsto$	$\{q p_2^l \pi_2^l\}$
you	$\mapsto$	$\{\pi_2\}$
seen	$\mapsto$	$\{p_2 o^l\}$

2. Add types for words with $\xrightarrow{GCONC^+}$: nothing

3. Rec. Calculus of types per segment, with $\xrightarrow{@}$

4. Test wether the sentence has atomic type s or $x \leq s$

Parsing algorithm in n^3

1. Types for words   2. + types to words with $\xrightarrow{GCONC^+}$

3. Rec. Calculus of types per segment, with $\xrightarrow{@}$

- Length = 1:

whom	have	you	seen
$\{q'o^{ll}q^l\}$	$\{qp_2^l\pi_2^l\}$	$\{\pi_2\}$	$\{p_2o^l\}$

- Length = 2:

whom have	have you	you seen
$\emptyset$	$\{qp_2^l\}$	$\emptyset$

- Length = 3:

whom have you	have you seen
$\{q'o^{ll}p_2^l\}$	$\{qo^l\}$

- Length = 4:

whom have you seen
$\{q'$ and $q'o^{ll}o^l\}$

4. Test wether the sentence has atomic type s or $x \leq s$

Parsing algorithm in n^3

1. Types for words
2. + types to words with $\xrightarrow{GCONC^+}$
3. Rec. Calculus of types per segment, with $\xrightarrow{@}$

- Length = 1:

whom	have	you	seen
$\{q'o^{ll}q^l\}$	$\{qp_2^l\pi_2^l\}$	$\{\pi_2\}$	$\{p_2o^l\}$

- Length = 2:

whom have	have you	you seen
$\emptyset$	$\{qp_2^l\}$	$\emptyset$

- Length = 3:

whom have you	have you seen
$\{q'o^{ll}p_2^l\}$	$\{qo^l\}$

- Length = 4:

whom have you seen
$\{q'$ and $q'o^{ll}o^l\}$

4. Test wether the sentence has atomic type s or $x \leq s$:
   $q' \in$ and $q' \leq s$

PG with ? and * : proposal

Weakening

$$\boxed{\frac{XY \leq Z}{Xp^{*(2k+1)}Y \leq Z}\,(* - W_L)} \qquad \frac{X \leq YZ}{X \leq Yp^{*(2k)}Z}\,(* - W_R)$$

Contraction

$$\boxed{\frac{Xp^{*(2k+1)}p^{(2k+1)}Y \leq Z}{Xp^{*(2k+1)}Y \leq Z}\,(* - C_L)} \qquad \frac{X \leq Yp^{(2k)}p^{*(2k)}Z}{X \leq Yp^{*(2k)}Z}\,(* - C_R)$$

$$\frac{Xp^{(2k+1)}p^{*(2k+1)}Y \leq Z}{Xp^{*(2k+1)}Y \leq Z}\,(* - C'_L) \qquad \frac{X \leq Yp^{*(2k)}p^{(2k)}Z}{X \leq Yp^{*(2k)}Z}\,(* - C'_R)$$

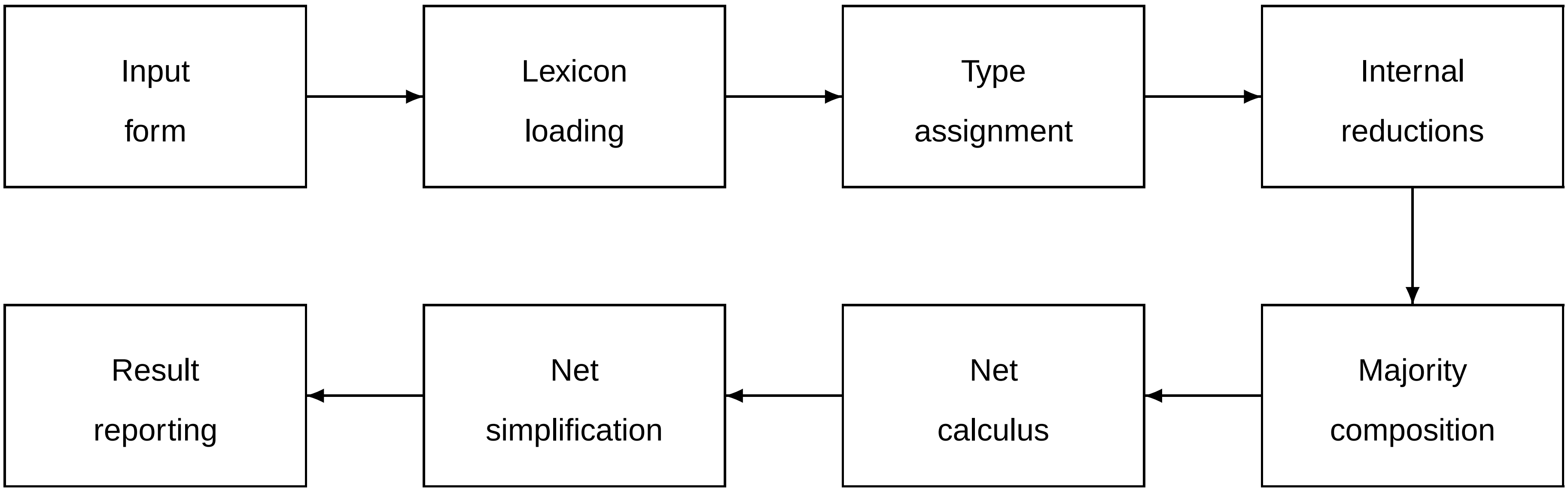
PG with ? and * : properties

- Property.[Optional and Iterated Basic Types]
  For a, a basic type:

$$a \leq a^? \qquad a^*a \leq a^*$$
$$1 \leq a^? \qquad aa^* \leq a^*$$
$$1 \leq a^*$$

- Theorem.The extended calculus defines a pregroup that extends the free pregroup based on $(P, \leq)$.
- Theorem.[The Cut Elimination] The cut rule can be eliminated in the extended calculus: every derivable inequality has a cut-free derivation.
- Property.[Decidability]
  The provability of $X \leq Y$ in this system is decidable

PPQ - overview



PPQ - grammar files

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar>
 <pregroup>
  <order inf="n" sup="n-bar"/>
...
 </pregroup>
 <sentence type="s"/>
 <lexicon>
  <w><word>whom</word>
     <type><simple atom="q’"/>
           <simple atom="o" exponent="-2"/>
           <simple atom="q" exponent="-1"/>
     </type>
  </w>
...
 </lexicon>
</grammar>
```

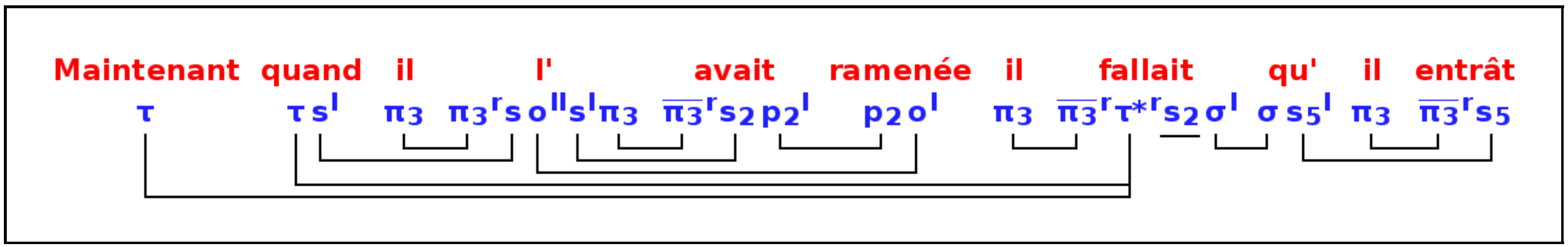
Lefff 2.5.5: 534753 entries $\Longrightarrow$ SQLite database lexicon.

PPQ - majority composition

The Matrix Content

cell 1-4 q'			
cell 1-3 $q'\ o^{ll}\ p\_2^{l}$	cell 2-4 $q\ o^{l}$		
cell 1-2	cell 2-3 $q\ p\_2^{l}$	cell 3-4	
cell 1-1 $q'\ o^{ll}\ q^{l}$	cell 2-2 $q\ p\_2^{l}\ \pi\_2^{l}$	cell 3-3 π_2	cell 4-4 $p\_2\ o^{l}$
whom	have	you	seen

PPQ - net calculus



[FR: *Now, when he took back her, he ought to enter*]

PPQ can output an XML representation of nets if the result must be used by another program

PPQ - net simplification

Maintenant	quand	il	l'	avait	ramenée	il	fallait	qu'	il	entrât
τ	$\tau\, s^l$	π_3	$\pi_3^r\, s\, o^{ll}\, s^l\, \pi_3$	$\overline{\pi_3}^r\, s_2\, p_2^l$	$p_2\, o^l$	π_3	$\overline{\pi_3}^r\, \tau^{*r}\, s_2\, \sigma^l$	$\sigma\, s_5^l$	π_3	$\overline{\pi_3}^r\, s_5$

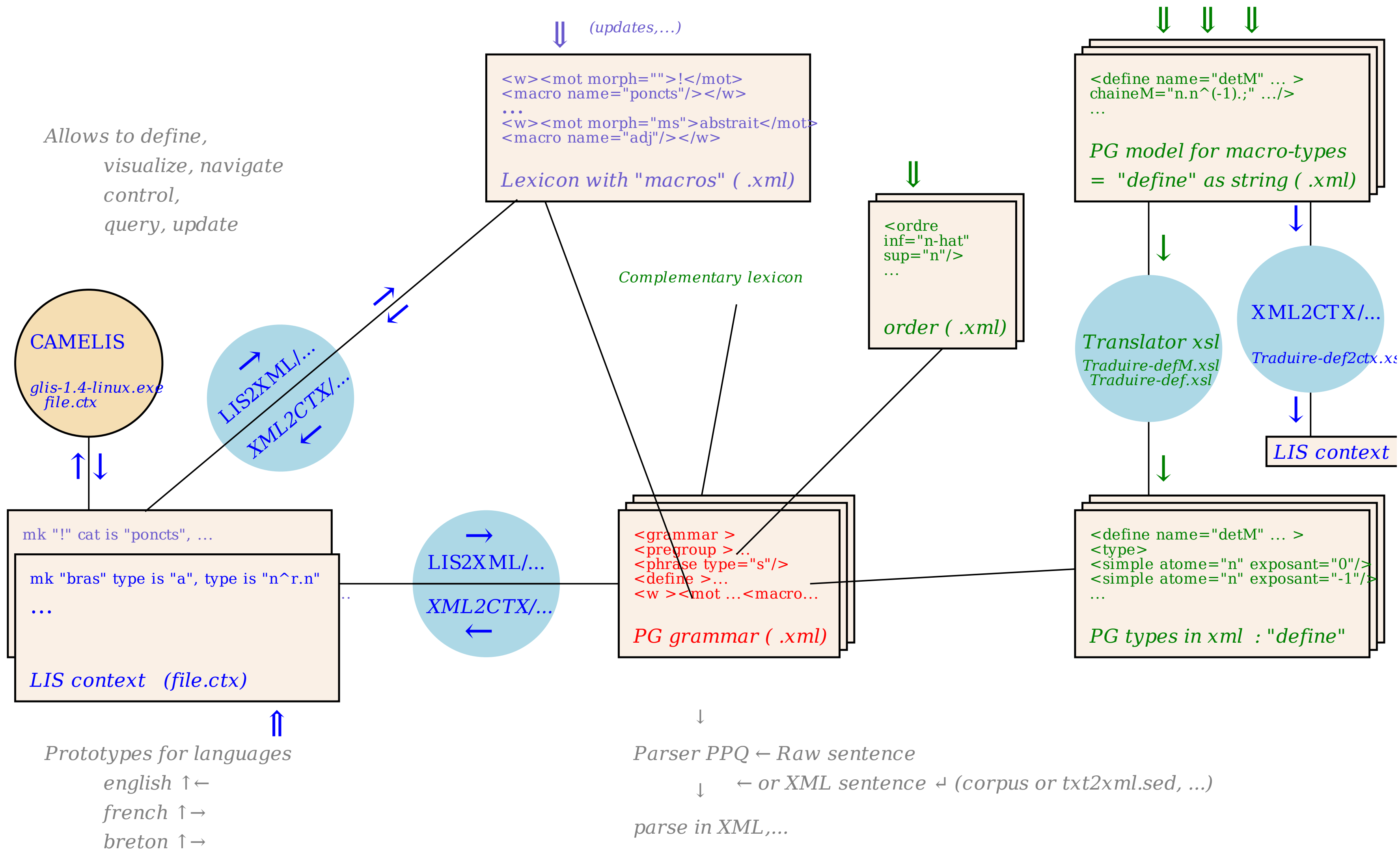
becomes

Maintenant	quand	il	l'	avait	ramenée	il	fallait	qu'	il	entrât
τ	$\tau\, s^l$	π_3	o^{ll}	$\overline{\pi_3}^r\, s_2\, p_2^l$	$p_2\, o^l$	π_3	$\overline{\pi_3}^r\, \tau^{*r}\, s_2\, \sigma^l$	$\sigma\, s_5^l$	π_3	$\overline{\pi_3}^r\, s_5$

[FR: *Now, when he took back her, he ought to enter*]

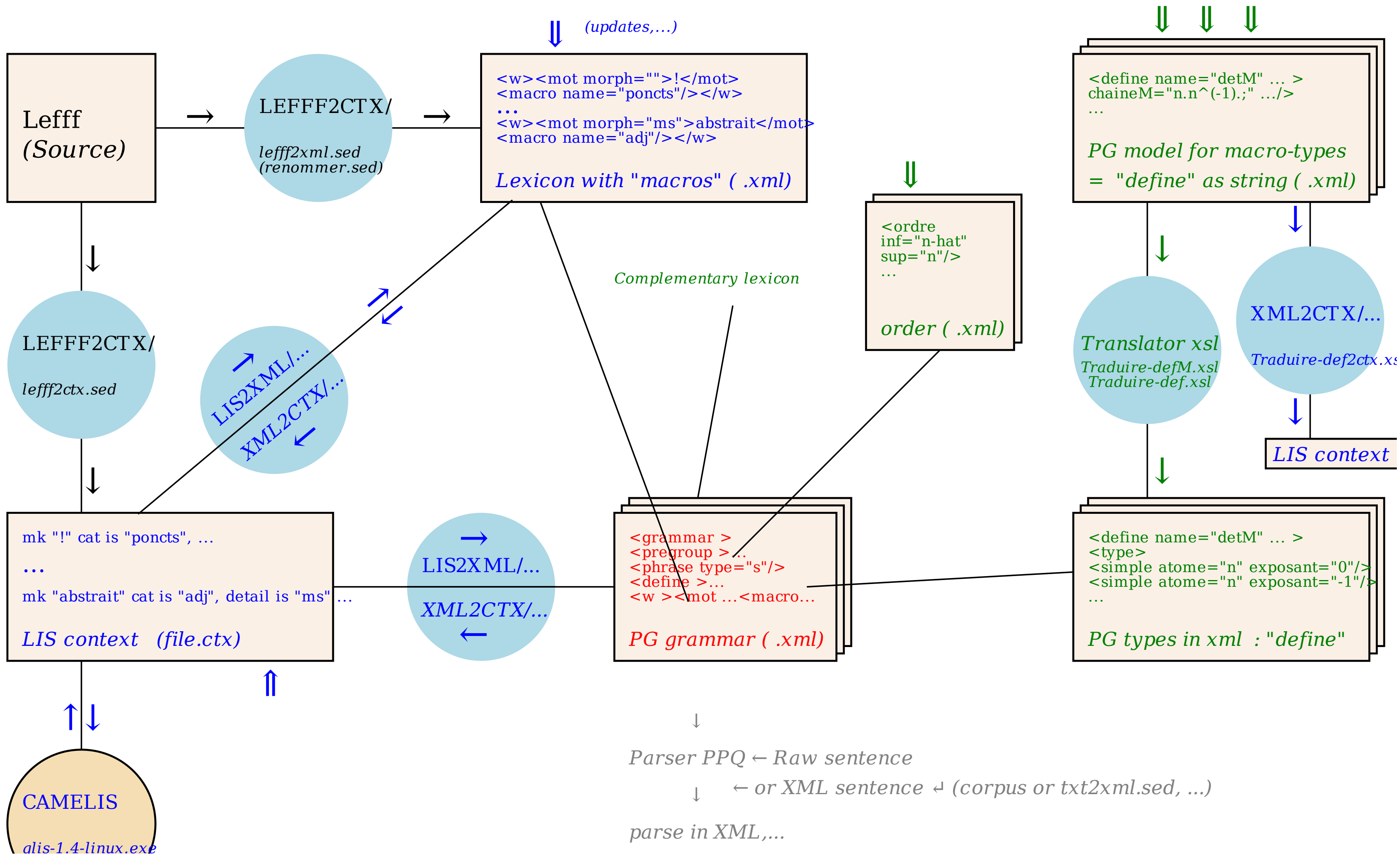
Grammar construction

Diagram : construction of PG grammars (with or without "macro-types") and use of CAMELIS



Grammar construction

Diagrams : around Lefff towards PG grammars (via "macro-types")



Conclusion

Parser using majority composition – a tool for experiments:

- Learning Categorial Grammars (Learnability of Pregroup Grammars [Studia Logica 87(2/3) 2007])
- Allow parsing that follows a partial tree (XML input) and can label subparts of sentences (like named entities)
- To test different ideas, including :
  - extensions of pregroups (*, ?) [LATA 2008]
  - "long distant dependencies" [To appear 2009]
  - different type assignment styles and languages
  - pregroup net sorting and filtering
  - ...

Small demo...