Non-projective axioms for pregroup grammars as cut eliminations

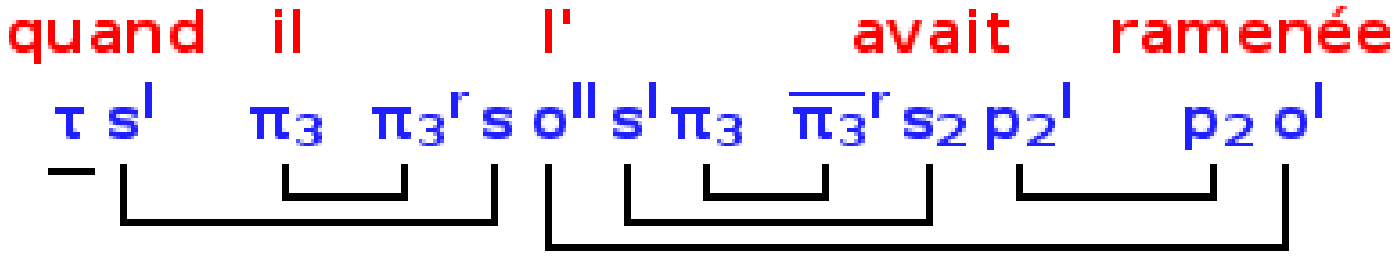
Denis Béchet

Denis.Bechet@univ-nantes.fr

LINA, University of Nantes

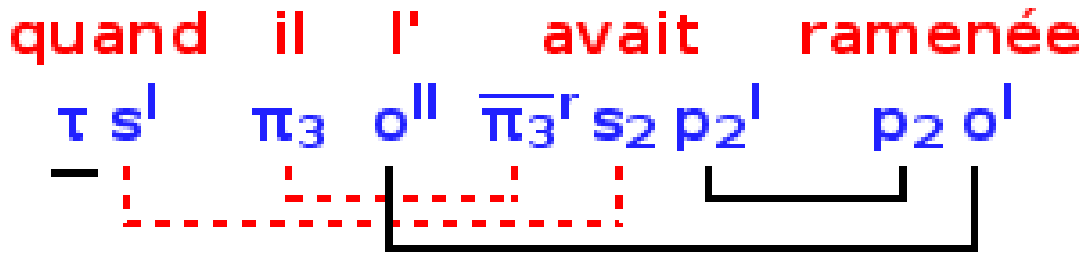
Non-projective axioms with pregroup

Pregroup analysis of “quand il l’avait ramené...” (when he took her home...):



l’ is assigned $\pi_3^r s o^{ll} s^l \pi_3$ rather than o^{ll}

A better analysis :



$\Longrightarrow$ We need non-projective axioms

Non-projective axioms with pregroup

How can we introduce “non projective axioms” ?

1. Using products of free pregroups (Kobele 2005)
   $\Longrightarrow$ NP complete (proof here)
2. Using an external product as with Categorial Dependency Grammars (Dikovsky 2004) (demo here)
   $\Longrightarrow$ Not a pregroup
3. Using “cut elimination”, non-projective axioms are created from projective axioms (demo here)
   $\Longrightarrow$ Complex annotation system and less powerful

Free pregroup

$$\frac{}{p^{(n)} \leq p^{(n)}}\,(Id) \qquad \frac{X \leq Y \quad Y \leq Z}{X \leq Z}\,(CUT)$$

$$\frac{XY \leq Z}{Xp^{(n)}p^{(n+1)}Y \leq Z}\,(A_G) \qquad \frac{X \leq YZ}{X \leq Yp^{(n+1)}p^{(n)}Z}\,(A_D)$$

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

$q \leq_{Pr} p$ if k is even or $p \leq_{Pr} q$ if k is odd

Pregroup grammars and languages

- A grammar $G = (\Sigma, (Pr, \leq_{Pr}), I, s)$:
  - Σ a finite alphabet
  - $(Pr, \leq)$ a finite partially ordered set (primitive types) that defines free pregroup $(Tp, \leq_{Tp})$
  - $I \subset \Sigma \times Tp$, a lexicon, assigns a finite set of types to each $c \in \Sigma$
  - $s \in Pr$ is a primitive type for correct sentences
- The language $\mathcal{L}(G) \in \Sigma^+$:

$$v_1 \cdots v_n \in \mathcal{L}(G)$$

iff

for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$ such that $X_1 \cdots X_n \leq_{Tp} s$

Can be adapted to any pregroup (not only free pregroup)

(not free) Pregroup grammars

- A grammar $G = (\Sigma, P, I, s)$:
  - Σ a finite alphabet
  - $P = (Tp, \bullet, 1, \leq_{Tp}, ^l, ^r)$ a (not free) pregroup
  - $I \subset \Sigma \times Tp$, a lexicon, assigns a finite set of types to each $c \in \Sigma$
  - $s \in Tp$ is a type for correct sentences
- The language $\mathcal{L}(G) \in \Sigma^+$:

$$v_1 \cdots v_n \in \mathcal{L}(G)$$

iff

for $1 \leq i \leq n$, $\exists X_i \in I(v_i)$ such that $X_1 \bullet \cdots \bullet X_n \leq_{Tp} s$

Non-projective axioms with pregroup

How can we introduce “non projective axioms” ?

1. Using products of free pregroups (Kobele 2005)
   $\Longrightarrow$ NP complete (proof here)
2. Using an external product as with Categorial Dependency Grammars (Dikovsky 2004) (demo here)
   $\Longrightarrow$ Not a pregroup
3. Using “cut elimination”, non-projective axioms are created from projective axioms (demo here)
   $\Longrightarrow$ Complex annotation system and less powerful

Product of pregoups (Kobele 2005)

For 2 pregroups $P_1 = (Tp_1, \bullet_1, 1_1, \leq_{Tp_1}, ^{l_1}, ^{r_1})$ and $P_2 = (Tp_2, \bullet_2, 1_2, \leq_{Tp_2}, ^{l_2}, ^{r_2})$:

$P_1 \times P_2 = (Tp_1 \times Tp_2, \circ, (1_1, 1_2), \leq, ^{L}, ^{R})$

1. $(x, y) \leq (x', y')$ iff $x \leq_{Tp_1} x'$ and $y \leq_{Tp_2} y'$
2. $(x, y) \circ (x', y') = (x \bullet_1 x', y \bullet_2 y')$
3. $(x, y)^L = (x^{l_1}, y^{l_2})$ and $(x, y)^R = (x^{r_1}, y^{r_2})$

Kobele 2005 : “Cross product” over lexicalized grammars

$$L(G_1 \times G_2) = L(G_2) \cap L(G_1)$$

But, grammars using products of free pregroups are NP complete (proof here)

Product of pregroup : NP hard

Proof : we encode any SAT problem

The proof uses the product of three free pregroups on $\{t, f\}$: (P, Δ_1, Δ_2) (in fact $P = \Delta_1 = \Delta_2$)

- P : for formula inferences
- Δ_1 and Δ_2 for the propagation of the boolean values of variables

A formula is transformed into a string.
The formula can be satisfied iff the string is in the language generated by a fixed grammar $\mathcal{G}_{SAT}$ based on $(P, Delta_1, Delta_2)$

NP hard (formula transformation)

A formula F that contains (at most) n variables $v_1, \ldots, v_n$ is transformed into a string $\mathcal{T}_n(F)$:

- $\mathcal{T}_n(F) = \underbrace{a \cdots a}_{n}[F]\underbrace{e \cdots e}_{n}$
- $[v_i]_n = \underbrace{b \cdots b}_{i-1} c \underbrace{b \cdots b}_{n-i} \underbrace{d \cdots d}_{n}$
- $[F_1 \vee F_2]_n = \vee [F_1]_n [F_2]_n$
- $[F_1 \wedge F_2]_n = \wedge [F_1]_n [F_2]_n$
- $[\neg F]_n = \neg [F]_n$

For instance : $\mathcal{T}_2(v_1 \vee (v_1 \wedge v_2)) = aa \vee \underbrace{cbdd}_{for\ v_1} \wedge \underbrace{cbdd}_{for\ v_1} \underbrace{bcdd}_{for\ v_2} ee$

NP hard (grammar)

$\mathcal{G}_{SAT}$ is defined by the following lexicon :

- $a \mapsto (1, t^l, 1) \text{or} (1, f^l, 1)$
- $b \mapsto (1, t, t^l) \text{or} (1, f, f^l)$
- $c \mapsto (t, t, t^l) \text{or} (f, f, f^l)$
- $d \mapsto (1, t^l, t) \text{or} (1, f^l, f)$
- $e \mapsto (1, t, 1) \text{or} (1, f, 1)$
- $\wedge \mapsto (tt^l t^l, 1, 1) \text{or} (f f^l t^l, 1, 1) \text{or} (f t^l f^l, 1, 1) \text{or} (f f^l f^l, 1, 1)$
- $\vee \mapsto (tt^l t^l, 1, 1) \text{or} (t f^l t^l, 1, 1) \text{or} (t t^l f^l, 1, 1) \text{or} (f f^l f^l, 1, 1)$
- $\neg \mapsto (t f^l, 1, 1) \text{or} (f t^l, 1, 1)$

The types corresponding to correct strings are $\leq (t, 1, 1)$

NP hard (example 1)

$$\mathcal{T}_1(v_1 \wedge v_1) = a \wedge cd\ cd\ e$$

For $v_1 = t$, the formula is true.

There exists an assignement of the symbols of $\mathcal{T}_1(v_1 \wedge v_1)$ through $\mathcal{G}_{SAT}$ whose product is $\leq (t, 1, 1)$:

$$\underbrace{(1, t^l, 1)}_{for\ a}\ \underbrace{(tt^l t^l, 1, 1)}_{for\ \wedge}\ \underbrace{(t, t, t^l)}_{for\ c}\ \underbrace{(1, t^l, t)}_{for\ d}\ \underbrace{(t, t, t^l)}_{for\ c}\ \underbrace{(1, t^l, t)}_{for\ d}\ \underbrace{(1, t, 1)}_{for\ e}$$

$$\leq (t, 1, 1)$$

NP hard (example 2)

$$\mathcal{T}_2(v_1 \wedge \neg v_2) = aa \wedge cbdd \neg bcdd\ ee$$

For $v_1 = t$ and $v_2 = f$, the formula is true.

There exists an assignement of the symbols of $\mathcal{T}_2(v_1 \wedge \neg v_2)$ through $\mathcal{G}_{SAT}$ whose product is $\leq (t, 1, 1)$:

$$\underbrace{(1, f^l, 1)}_{for\ a}\underbrace{(1, t^l, 1)}_{for\ a}\underbrace{(tt^lt^l, 1, 1)}_{for\ \wedge}\underbrace{(t, t, t^l)}_{for\ c}\underbrace{(1, f, f^l)}_{for\ b}\underbrace{(1, f^l, f)}_{for\ d}\underbrace{(1, t^l, t)}_{for\ d}$$

$$\underbrace{(tf^l, 1, 1)}_{for\ \neg}\underbrace{(1, t, t^l)}_{for\ b}\underbrace{(f, f, f^l)}_{for\ c}\underbrace{(1, f^l, f)}_{for\ d}\underbrace{(1, t^l, t)}_{for\ d}\underbrace{(1, t, 1)}_{for\ e}\underbrace{(1, f, 1)}_{for\ e}$$

$$\leq (t, 1, 1)$$

Product of pregroup : NP complete

A grammar using the product of N free pregroups is in NP because, to test if a string is in the language, we just have to know the right assignment through the grammar and test that the N pregroup type components are less (or equal) than the corresponding pregroup type components associated to correct sentences.

$\Rightarrow$ Product of N free pregroups for $N \geq 3$ is NP complete

Remark: This is also true for $N = 2$ (the proof is similar but the construction is more complex)

Non-projective axioms with pregroup

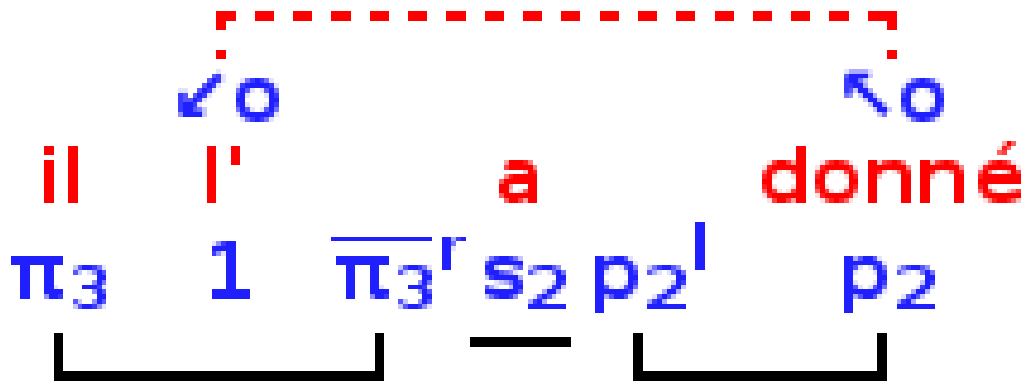
How can we introduce “non projective axioms” ?

1. Using products of free pregroups (Kobele 2005)
   $\Longrightarrow$ NP complete (proof here)
2. Using an external product as with Categorial Dependency Grammars (Dikovsky 2004) (demo here)
   $\Longrightarrow$ Not a pregroup
3. Using “cut elimination”, non-projective axioms are created from projective axioms (demo here)
   $\Longrightarrow$ Complex annotation system and less powerful

External product with a free pregroup

Is it possible to have a product of a free pregroup with a simpler structure (i.e. with a polynomial complexity) ?

$\Rightarrow$ Pregroup with potential



Free pregroup with potential

Structure like Categorial Dependency Grammar (Dikovsky 2004) but with a free pregroup rather than a set of flat categorial types :

$P \times \Delta_1 \times \cdots \times \Delta_n$

- P : a (free) pregroup used by the grammar as a pregroup grammar
- Δ_i : strings of parentheses used by the grammar as a Dyck language (with only one couple of parentheses)

Proprety :

- Parsing is polynomial
- Some languages are context-sensitive languages

Problems :

- Strings of parentheses do not form a pregroup
- We need to introduce “anchors”

Free pregroup with potential

Strings of parentheses do not form a pregroup because :

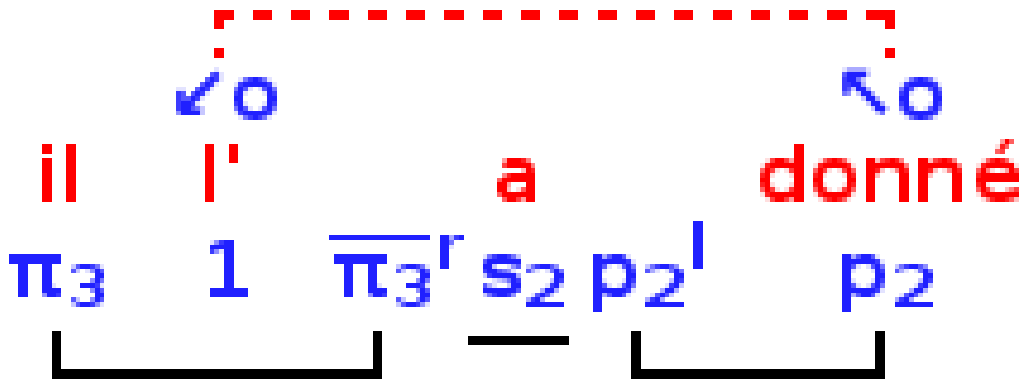
We need at the same time a left adjoint and a right adjoint for "$\nwarrow o$"

- If $(\nwarrow o)^l = (\nwarrow o)^r = \swarrow o$ then the structure is not a Dyck language
- If $(\nwarrow o)^l \neq (\nwarrow o)^r$ then the structure has at least three generators

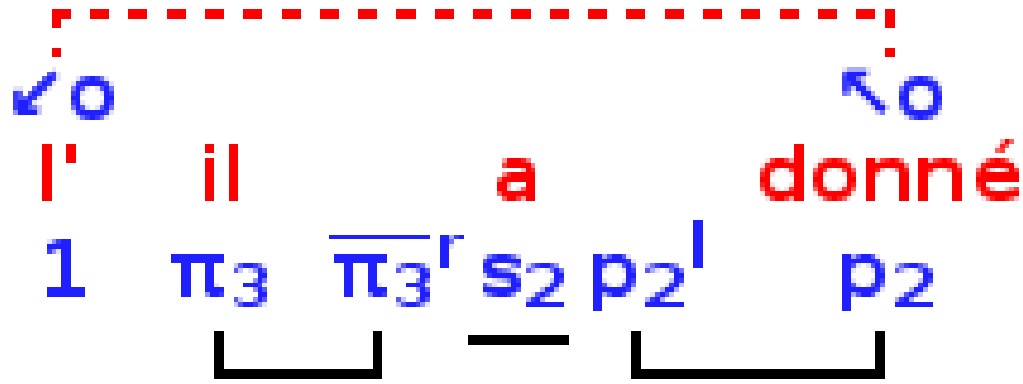
$\Longrightarrow$ A free pregroup with potential is not a pregroup

Free pregroup with potential

We need to introduce “anchors” to control the end position of a “non projective axiom”

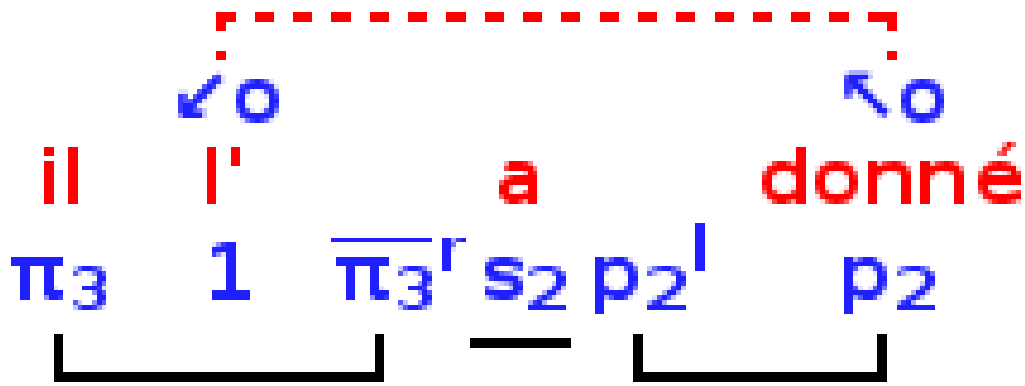


Accept also :

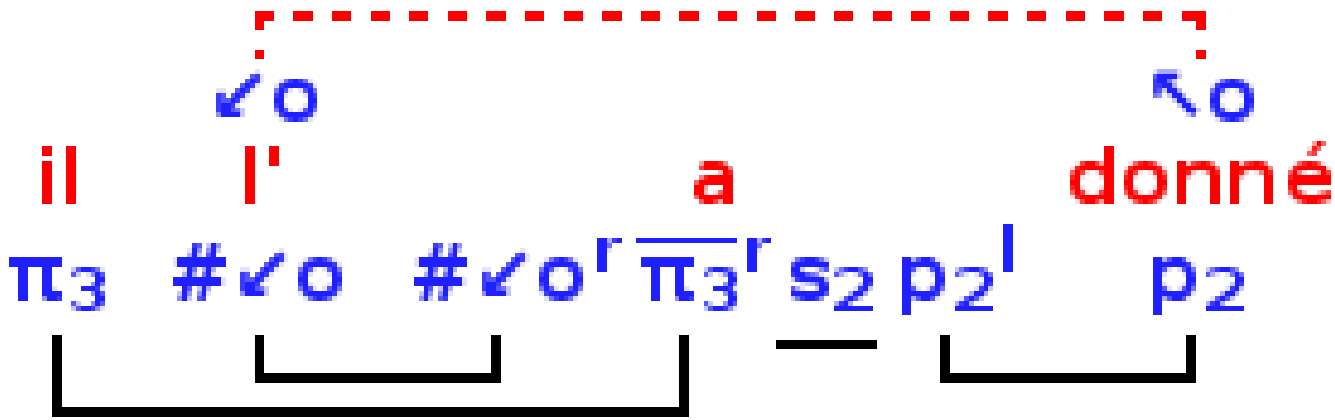


Free pregroup with potential

Without “anchors” :



With “anchors” :



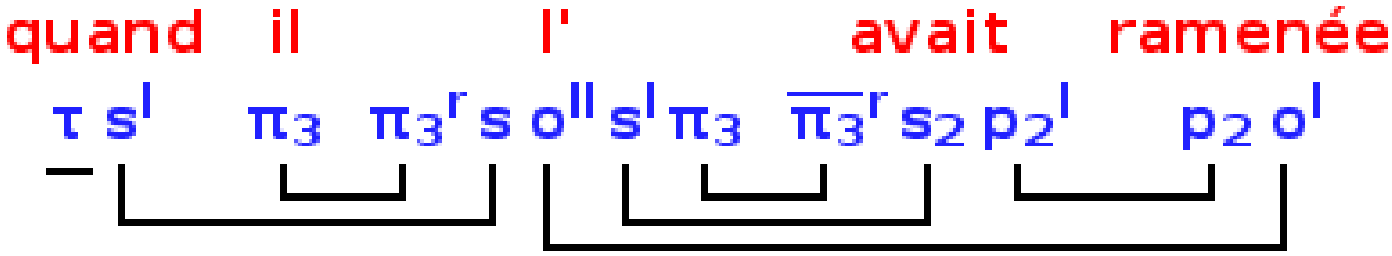
Non-projective axioms with pregroup

How can we introduce “non projective axioms” ?

1. Using products of free pregroups (Kobele 2005)
   $\Longrightarrow$ NP complete (proof here)
2. Using an external product as with Categorial Dependency Grammars (Dikovsky 2004) (demo here)
   $\Longrightarrow$ Not a pregroup
3. Using “cut elimination”, non-projective axioms are created from projective axioms (demo here)
   $\Longrightarrow$ Complex annotation system and less powerful

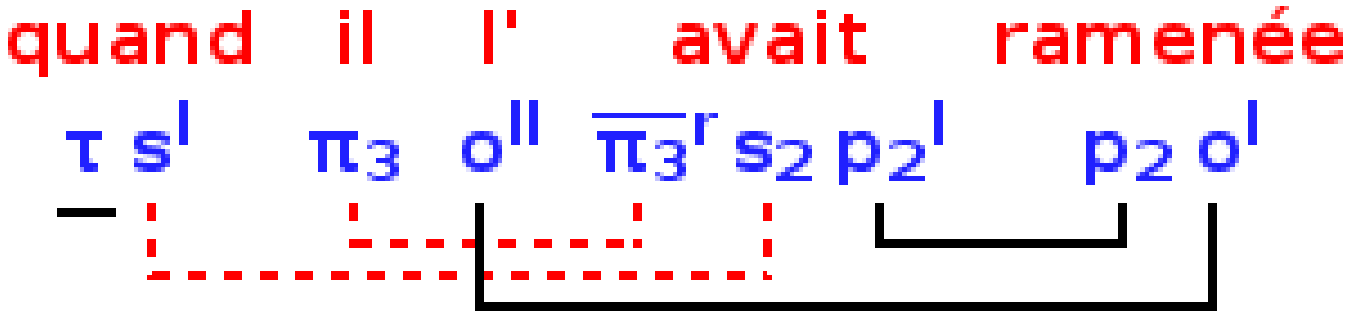
Free pregroup with cut annotations

We introduce cut annotations $[\cdots]$ on types

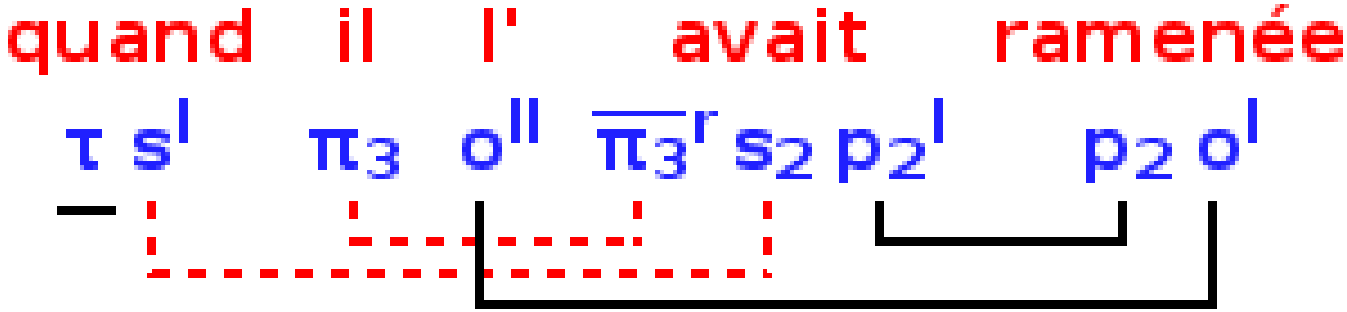


$$l' \mapsto [\pi_3^r [s o^{ll} s^l] \pi_3]$$

After "annotated cut elimination" of $[\pi_3^r [s o^{ll} s^l] \pi_3]$:



Free pregroup with cut annotations



- Non-projective axioms introduced at a final step
- The complexity is polynomial
- Does not extend the class of languages
- Needs (not natural) annotations on types

Conclusion

- Non-projective axioms in pregroup are not easy to introduce
- Here, 3 propositions (2 are implemented) that are not completely satisfying :
  - Must be polynomial
  - Uses a (not free) pregroup
  - Extends the class of languages beyong the class of context free languages

Better solution ?