Categorial Dependency Grammars: from Theory to Large Scale Grammars

A. Dikovsky (LINA, CNRS UMR 6241), Université de Nantes

Depling'2011, Barselona, September 5-7, 2011

Matter of Concern

1. The matter concerns Categorial Dependency Grammars (CDG) [Dikovsky'2004] :
   - fully lexicalized, rule-less type based D-grammars
   - expressing unlimited non-projective dependencies
   - and parsed in polynomial time

Matter of Concern

1. The matter concerns Categorial Dependency Grammars (CDG) [Dikovsky'2004] :
   - fully lexicalized, rule-less type based D-grammars
   - expressing unlimited non-projective dependencies
   - and parsed in polynomial time
2. We show a practical method of incremental bootstrapping of wide scope large scale CDG from examples of correct dependency structures

Plan

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies

A CDG is an assignment of type sets to words :

$$w \mapsto [d_1 \backslash \ldots \backslash d_0 / \ldots d_n]^{\{\ldots v \ldots\}}$$

Intuitively, this assignment defines dependencies of w :

PROJECTIVE DEPENDENCIES

Dependency : $Gov \xrightarrow{d} Sub$:

Governor Type : $Gov \mapsto [..\backslash../../d/..]$

Subordinate Type : $Sub \mapsto [..\backslash d/..]$

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies

NON-PROJECTIVE DEPENDENCIES

Polarized valencies : $\nearrow d$, $\searrow d$, $\nwarrow d$, $\swarrow d$

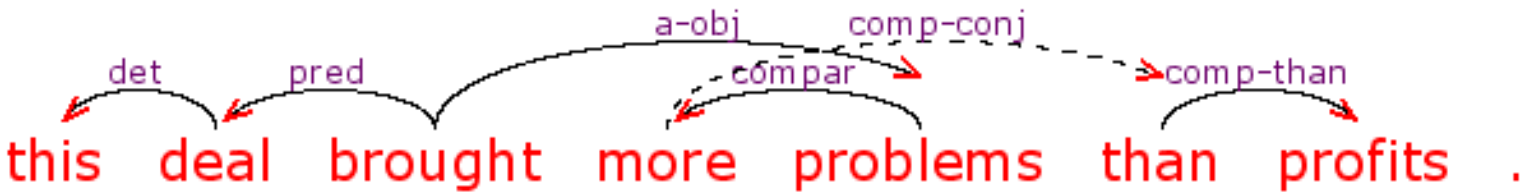
Dependency : $Gov \overset{d}{--}> Sub$:

Governor Type Potential : $Gov \mapsto [..]^{\nearrow d..}$

Subordinate Type Potential : $Sub \mapsto [..]^{\searrow d..}$

Categorial Dependency Grammars (CDG)

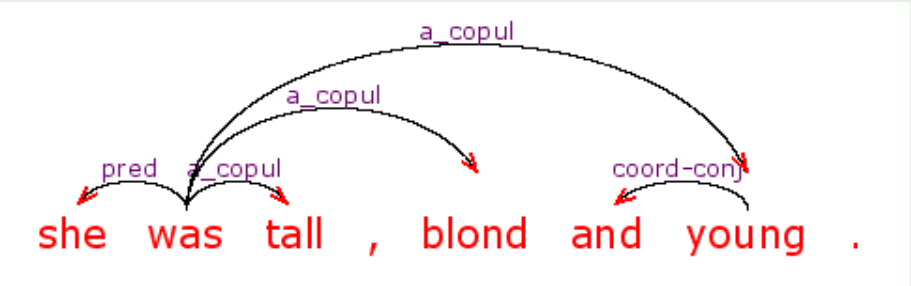
CDG Types express dependency valencies



$this \mapsto [det]$
$deal \mapsto [det \backslash pred]$
$brought \mapsto [pred \backslash S / a-obj]$
$problems \mapsto [compar \backslash a-obj]$
$profits \mapsto [comp-than]$
$more \mapsto [compar]^{\nearrow comp\text{–}conj}$
$than \mapsto [/comp-than]^{\searrow comp\text{–}conj}$

Repeatable dependencies

Some dependency valencies are MULTIPLE

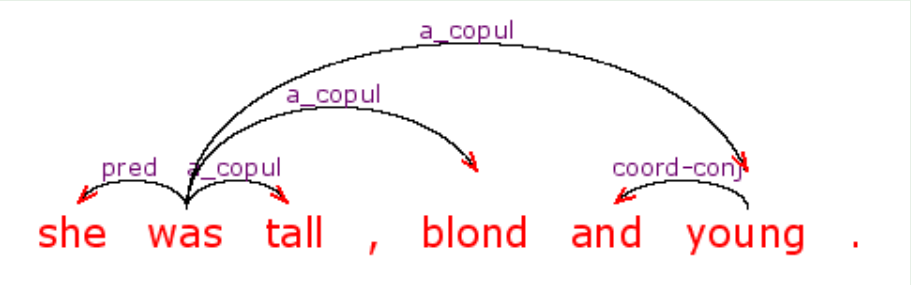


pred is non-repeatable

a_copul is repeatable

Repeatable dependencies

Some dependency valencies are MULTIPLE



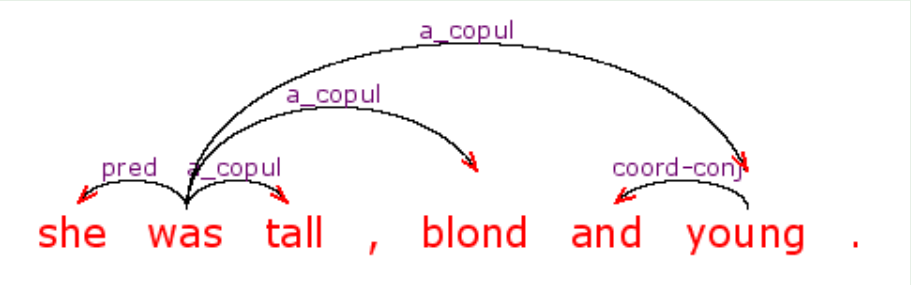
pred is non-repeatable
a_copul is repeatable

Principle of Repeatable Dependencies [Mel'čuk'88]

- every dependency d is either repeatable or non-repeatable;
- d is repeatable if SOME governor uses d in SOME DS at least ($K =$) 2 times;
- any word governing through a repeatable dependency d in SOME DS may have any number of subordinates through d

Repeatable dependencies

Some dependency valencies are MULTIPLE



pred is non-repeatable
a_copul is repeatable

Principle of Repeatable Dependencies [Mel'čuk'88]

- every dependency d is either repeatable or non-repeatable;
- d is repeatable if SOME governor uses d in SOME DS at least ($K =$) 2 times;
- any word governing through a repeatable dependency d in SOME DS may have any number of subordinates through d

Iterated subtypes for repeatable dependencies

$$was \mapsto [pred \backslash S / a\_copul^*]$$

CDG calculus

Left-oriented rules

$\mathbf{L^l}.\quad C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1 P_2}$ $\qquad Gov \xrightarrow{C} Sub$

CDG calculus

Left-oriented rules

$\mathbf{L^l}$. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1 P_2}$ $\quad Gov \xrightarrow{C} Sub$

$\mathbf{I^l}$. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1 P_2}$ $\quad Gov \xrightarrow{C} Sub$

$\mathbf{\Omega^l}$. $[C^*\backslash\beta]^{P} \vdash [\beta]^{P}$

CDG calculus

Left-oriented rules

L^l. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1 P_2}$ $\quad Gov \xrightarrow{C} Sub$

I^l. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1 P_2}$ $\quad Gov \xrightarrow{C} Sub$

Ω^l. $[C^*\backslash\beta]^{P} \vdash [\beta]^{P}$

D^l. $\alpha^{P_1}(\swarrow C)P(\nwarrow C)^{P_2} \vdash \alpha^{P_1 P P_2}$ $\quad Gov \overset{C}{-\!-\!>} Sub$

"First-Available" Rule of pairing polarized valencies

FA : in $(\swarrow C)P(\nwarrow C)$, the valency $\swarrow C$ is the **first available** for the dual valency $\nwarrow C$, i.e. P has no occurrences of $\swarrow C$, $\nwarrow C$

Derivation Example

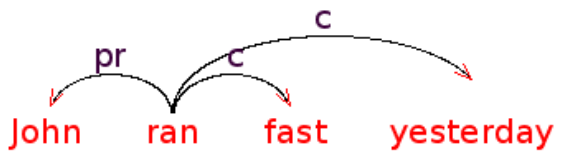
LEXICON :
(empty potentials)

$John \mapsto [pr]$
$ran \mapsto [pr \backslash S / c^*]$
$fast \mapsto [c]$

Derivation

$$\dfrac{\dfrac{\dfrac{\overset{John}{[pr]} \quad \dfrac{\overset{ran}{[pr \backslash S / c^*]} \quad \overset{fast}{[c]}}{[pr \backslash S / c^*]}\, \mathbf{I^r}}{[pr \backslash S]}\, \Omega^r \quad \overset{yesterday}{[c]}}{[pr \backslash S / c^*]}\, \mathbf{I^r}}{S}\, \mathbf{L^l}$$

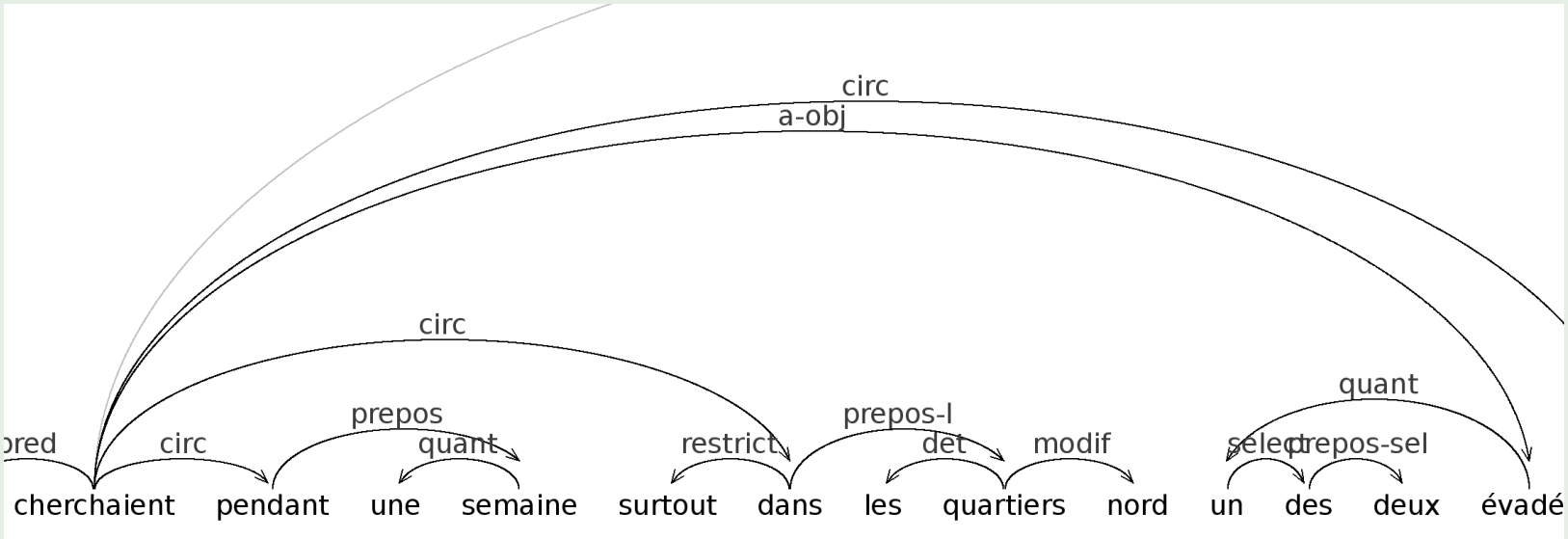
Dependency structures



$\mathbf{L^l}$	$H^{P_1}[H\backslash\beta]^{P_2} \vdash [\beta]^{P_1 P_2}$	$\mathbf{L^r}$	$[\beta \,/\, H]^{P_2} H^{P_1} \vdash [\beta]^{P_2 P_1}$
$\mathbf{I^l}$	$C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1 P_2}$	$\mathbf{I^r}$	$[\beta \,/\, C^*]^{P_2} C^{P_1} \vdash [\beta \,/\, C^*]^{P_2 P_1}$
Ω^l	$[C^*\backslash\beta]^{P} \vdash [\beta]^{P}$	Ω^r	$[\beta \,/\, C^*]^{P} \vdash [\beta]^{P}$

Non-sequential Repeatable Dependencies

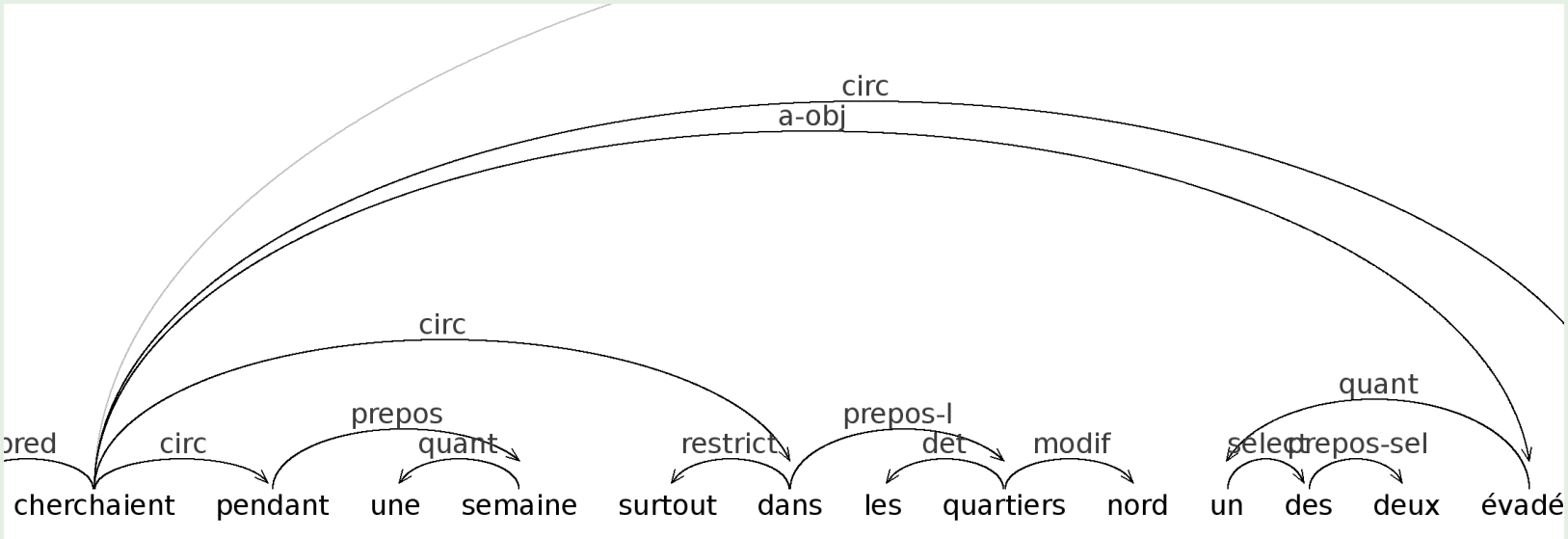
Word Order and Repeatability



"They tracked for a week especially in the nord quarters one of the two fugitives systematically blocking entries and exits"

Non-sequential Repeatable Dependencies

Word Order and Repeatability



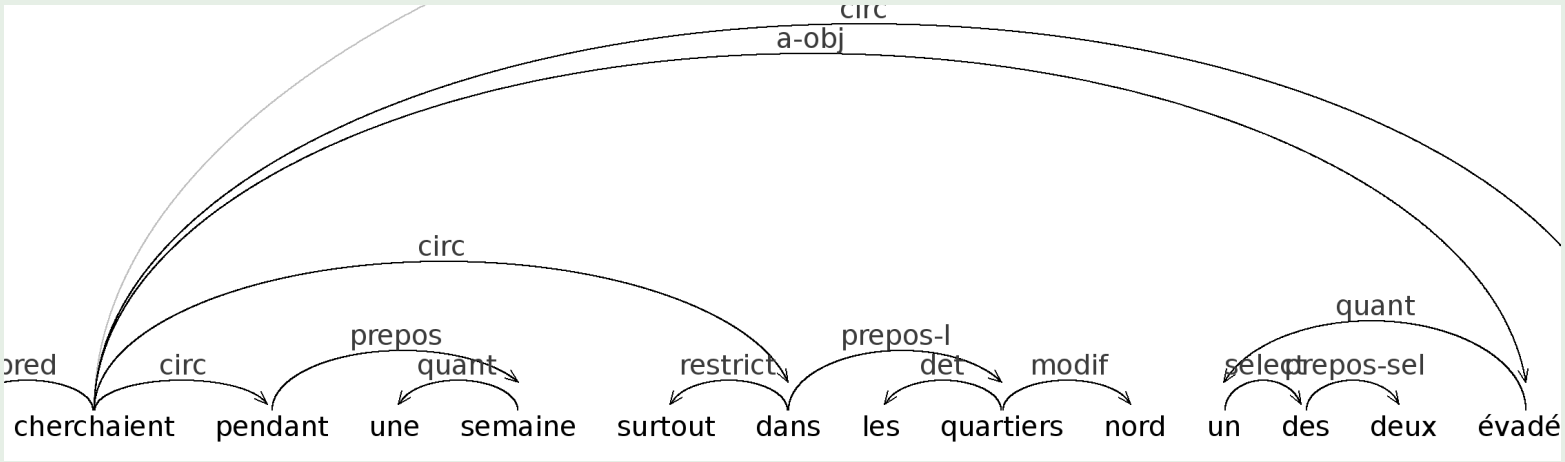
"They tracked for a week especially in the nord quarters one of the two fugitives systematically blocking entries and exits"

Repeatable dependency not expressed using iteration

Dependency circ is repeatable for $K = 3$, but it is not sequential

Dispersed Iteration Types

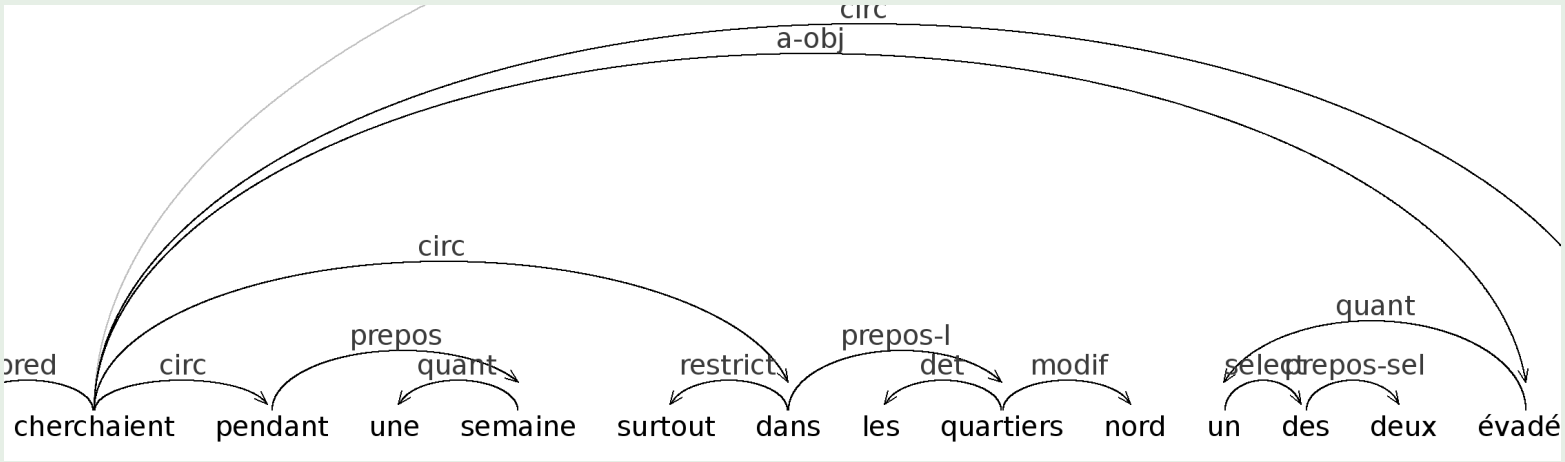
Dispersed Repeatable Dependencies ($K = 3$)



DISPERSED iteration : $cherchaient \mapsto [pred\backslash S/@fs/a-obj/\{circ^*\}]$

Dispersed Iteration Types

Dispersed Repeatable Dependencies ($K = 3$)



DISPERSED iteration : $cherchaient \mapsto [pred\backslash S/@fs/a-obj/\{circ^*\}]$

Extended Type Calculus

DISPERSED iteration

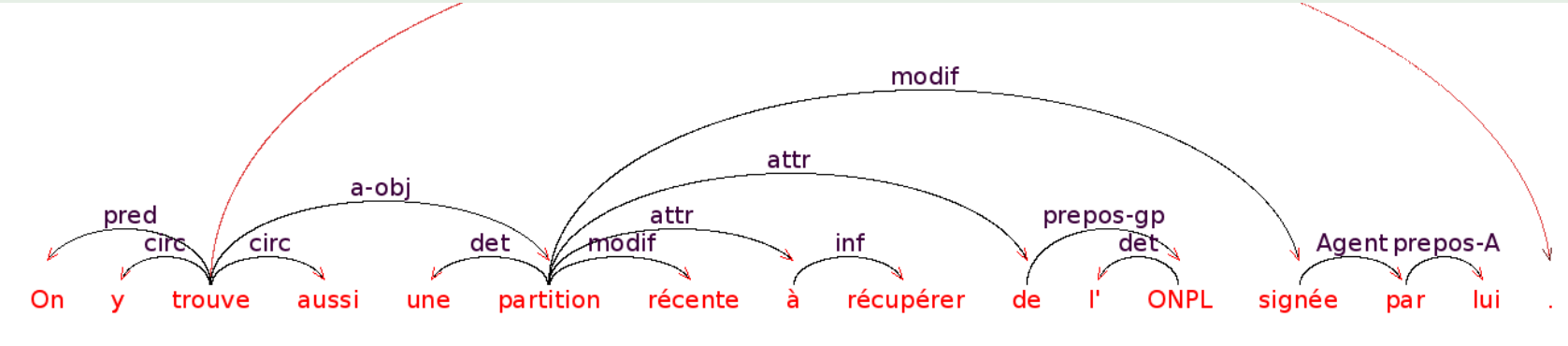
$$\mathbf{LD^I}.\quad C^{P_1}[\{\alpha_1\}\backslash C\backslash\beta/\{\alpha_2\}]^{P_2} \vdash [\{\alpha_1\}\backslash\beta/\{\alpha_2\}]^{P_1P_2}$$

$$\mathbf{ID^I}.\quad C^{P_1}[\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\alpha_3\}]^{P_2} \vdash [\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\alpha_3\}]^{P_1P_2}$$

$$\mathbf{\Omega C^I}.\quad [\{\alpha_1, C^*, \alpha_2\}\backslash\beta/\{\alpha_3\}]^{P} \vdash [\{\alpha_1, \alpha_2\}\backslash\beta/\{\alpha_3\}]^{P}$$

Choice Iteration Types

Choice Repeatable Dependencies ($K = 2$)

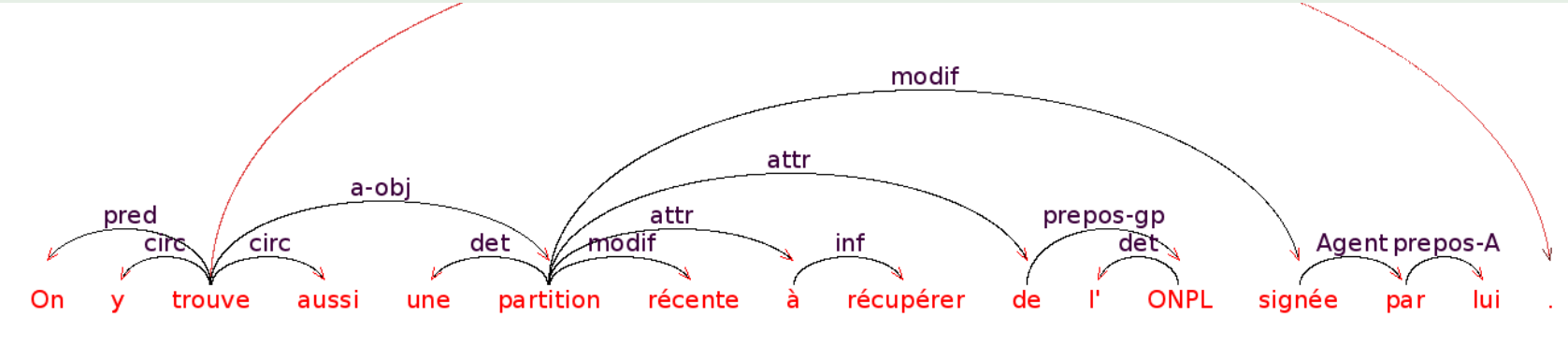


*One here finds also a partition recent to obtain of ONPL signed by him

CHOICE iteration : $partition \mapsto [det \backslash a-obj / (modif \,|\, attr)^*]$

Choice Iteration Types

Choice Repeatable Dependencies ($K = 2$)



*One here finds also a partition recent to obtain of ONPL signed by him

CHOICE iteration : $partition \mapsto [det\backslash a-obj/(modif|attr)^*]$

Extended Type Calculus

CHOICE iteration

LC$^{\mathsf{I}}$. $C^{P_1}[(\alpha_1|C|\alpha_2)\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$

IC$^{\mathsf{I}}$. $C^{P_1}[(\alpha_1|C|\alpha_2)^*\backslash\beta]^{P_2} \vdash [(\alpha_1|C|\alpha_2)^*\backslash\beta]^{P_1P_2}$

ΩC$^{\mathsf{I}}$. $[(\alpha_1|C|\alpha_2)^*\backslash\beta]^{P} \vdash [\beta]^{P}$

Extended CDG [Dikovsky'2009]

- use Regular Type Expressions (RTE) in place of types

- RTE extend the types by :
  Choice : $(prepos-g|prepos-d|prepos-l|prepos-o|prepos-A)$;
  Option : $deict?, (claus|loc-rel)?$;
  Choice iteration : $(attr|circ)*$;
  Dispersed subtypes expressing flexible order :
  Left/Right :
  $[\{compar?, modif*, restrict?\}\backslash det?\backslash quantif?\backslash emphat\backslash\#(\searrow coref)/(claus|loc-rel)?/attr*/g-obj?/\{appos?\}] <\searrow coref >$;
  Two-way :
  $\{(\#(\searrow explet)|explet)?\}[(restrict|det)?\backslash(pred|a-obj)/\#(\searrow cm)?/(attr|circ)*]$

Extended CDG [Dikovsky'2009]

- use Regular Type Expressions (RTE) in place of types

- RTE extend the types by :
  Choice : $(prepos-g|prepos-d|prepos-l|prepos-o|prepos-A)$;
  Option : $deict?, (claus|loc-rel)?$;
  Choice iteration : $(attr|circ)*$;
  Dispersed subtypes expressing flexible order :
  Left/Right :
  $[\{compar?, modif*, restrict?\}\backslash det?\backslash quantif?\backslash emphat\backslash\#(\searrow coref)/(claus|loc-rel)?/attr*/g-obj?/\{appos?\}] <\searrow coref >$;
  Two-way :
  $\{(\#(\searrow explet)|explet)?\}[(restrict|det)?\backslash(pred|a-obj)/\#(\searrow cm)?/(attr|circ)*]$
- RTE are FLAT, i.e. depth bounded (max. depth is 3) :
  $\{(restrict|quantif|emphat)?, \#(\searrow explet)?\}[junc\backslash\#(\searrow cm)?\backslash coordn/(loc-rel|claus)?/attr?/modif*/\{appos?\}]$

Extended CDG

- use Lexical Classes in place of words

Dictionary W is covered by classes : $W = \bigcup_{i \in I} C_i$ and the lexicon λ assigns sets of RTE to classes. At that :

- all words in a class C share the types defined by the RTE assigned to C
- every word has all types of the classes to which it belongs.

Extended CDG

- use Lexical Classes in place of words

Dictionary W is covered by classes : $W = \bigcup_{i \in I} C_i$ and the lexicon λ assigns sets of RTE to classes. At that :

- all words in a class C share the types defined by the RTE assigned to C
- every word has all types of the classes to which it belongs.

A class definition

$PN(Lex = quant, C = a|p)$
$\leftarrow$ nul, rien, tout
$\{explet?\}[(restrict|det)?\backslash(pred|a-obj)/\#(\searrow cm)?/(attr|circ)*]$
$[restrict?\backslash(prepos-g|qa-obj)]$

Properties of Extended CDG

Expressivity. Extended CDG are :

- equivalent to basic CDG
- more expressive than CF-grammars, e.g. they generate $L^{(m)} = \{a_1^n a_2^n ... a_m^n \mid n \geq 1\}$ for all $m > 0$ and $MIX = \{w \in \{a, b, c\}^+ \mid |w|_a = |w|_b = |w|_c\}$.
- equivalent to real time 1-stack automata with independent counters
- parsed in polynomial time using a practical tabular algorithm

Properties of Extended CDG

Expressivity. Extended CDG are :

- equivalent to basic CDG
- more expressive than CF-grammars, e.g. they generate $L^{(m)} = \{a_1^n a_2^n ... a_m^n \mid n \geq 1\}$ for all $m > 0$ and $MIX = \{w \in \{a, b, c\}^+ \mid |w|_a = |w|_b = |w|_c\}$.
- equivalent to real time 1-stack automata with independent counters
- parsed in polynomial time using a practical tabular algorithm

CDG and Extended CDG are not learnable from DS

Recently [Bechet,Dikovsky,Foret'2010-11] it was proved that :

- Unlimited CGD are not learnable from treebanks using the traditional unification based methods
- Important subclasses of Extended CDG respecting the Repeatable Dependency Principle are learnable from DS

Properties of Extended CDG

Expressivity. Extended CDG are :

- equivalent to basic CDG
- more expressive than CF-grammars, e.g. they generate $L^{(m)} = \{a_1^n a_2^n ... a_m^n \mid n \geq 1\}$ for all $m > 0$ and $MIX = \{w \in \{a, b, c\}^+ \mid |w|_a = |w|_b = |w|_c\}$.
- equivalent to real time 1-stack automata with independent counters
- parsed in polynomial time using a practical tabular algorithm

CDG and Extended CDG are not learnable from DS

Recently [Bechet,Dikovsky,Foret'2010-11] it was proved that :

- Unlimited CGD are not learnable from treebanks using the traditional unification based methods
- Important subclasses of Extended CDG respecting the Repeatable Dependency Principle are learnable from DS

In place of learning, we propose a practical incremental method of BOOTSTRAPPING Extended CDG from samples of DS

Genericity Order

PO $\lesssim$ (X : a list of alternatives ; $(t)=_{df}\ t$)

1. $t \lesssim (t|X)$
2. $(t|X) \lesssim (t|X)?$
3. $(t|X)? \lesssim (t|X)^*$
4. $\{\gamma\}[\{\gamma_1\}\backslash t\backslash\beta]^P \lesssim \{\gamma\}[\{t,\gamma_1\}\backslash\beta]^P$ (similar for right)
5. $\{\gamma\}[\{t,\gamma_1\}\backslash\beta]^P \lesssim \{t,\gamma\}[\{\gamma_1\}\backslash\beta]^P$
6. $\{\gamma\}[\alpha/\{t,\gamma_1\}]^P \lesssim \{t,\gamma\}[\alpha/\{\gamma_1\}]^P$.

Genericity Order

PO $\lesssim$ (X : a list of alternatives ; $(t)=_{df} t$)

1. $t \lesssim (t|X)$
2. $(t|X) \lesssim (t|X)?$
3. $(t|X)? \lesssim (t|X)^*$
4. $\{\gamma\}[\{\gamma_1\}\backslash t\backslash\beta]^P \lesssim \{\gamma\}[\{t,\gamma_1\}\backslash\beta]^P$ (similar for right)
5. $\{\gamma\}[\{t,\gamma_1\}\backslash\beta]^P \lesssim \{t,\gamma\}[\{\gamma_1\}\backslash\beta]^P$
6. $\{\gamma\}[\alpha/\{t,\gamma_1\}]^P \lesssim \{t,\gamma\}[\alpha/\{\gamma_1\}]^P$.

Genericity PO $\preceq$

$\preceq$ is the closure of $\lesssim$ by reflexivity, transitivity and by type construction. It is extendable to CDG through lexicons λ

Genericity Order

PO $\lesssim$ (X : a list of alternatives ; $(t)=_{df} t$)

1. $t \lesssim (t|X)$
2. $(t|X) \lesssim (t|X)?$
3. $(t|X)? \lesssim (t|X)^*$
4. $\{\gamma\}[\{\gamma_1\}\backslash t\backslash\beta]^P \lesssim \{\gamma\}[\{t,\gamma_1\}\backslash\beta]^P$ (similar for right)
5. $\{\gamma\}[\{t,\gamma_1\}\backslash\beta]^P \lesssim \{t,\gamma\}[\{\gamma_1\}\backslash\beta]^P$
6. $\{\gamma\}[\alpha/\{t,\gamma_1\}]^P \lesssim \{t,\gamma\}[\alpha/\{\gamma_1\}]^P$.

Genericity PO $\preceq$

$\preceq$ is the closure of $\lesssim$ by reflexivity, transitivity and by type construction. It is extendable to CDG through lexicons λ

$\preceq$ is monotone wrt DS-languages $\Delta(G)$:

$$G_1 \preceq G_2 \Rightarrow \Delta(G_1) \subseteq \Delta(G_2)$$

Idea of the Structural Bootstrapping Method (SBM)

SBM updates Extended CDG G_{in} for every input DS D

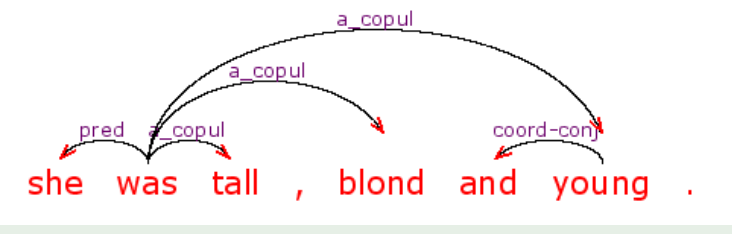
- if the vicinity $V(w, D_x)$ of a word w in D is generated by no RTE assigned to w in G_{in}
- then SBM finds a minimal more general Extended CDG G_{out}, i.e. $G_{in} \preceq G_{out}$, which generates $V(w, D)$

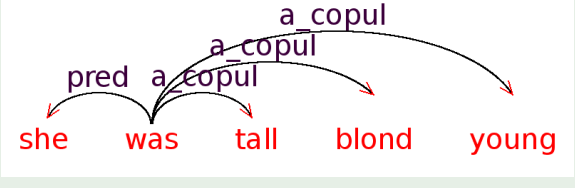
Idea of the Structural Bootstrapping Method (SBM)

SBM updates Extended CDG G_{in} for every input DS D

- if the vicinity $V(w, D_x)$ of a word w in D is generated by no RTE assigned to w in G_{in}
- then SBM finds a minimal more general Extended CDG G_{out}, i.e. $G_{in} \preceq G_{out}$, which generates $V(w, D)$

Vicinity Example : $V(was, D)$



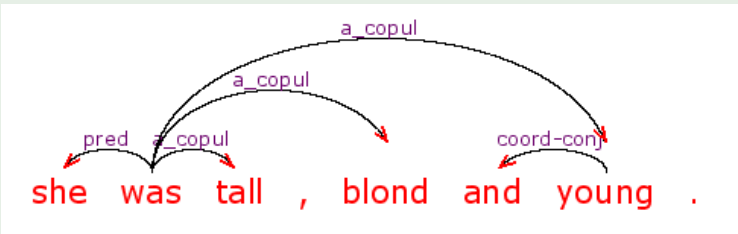


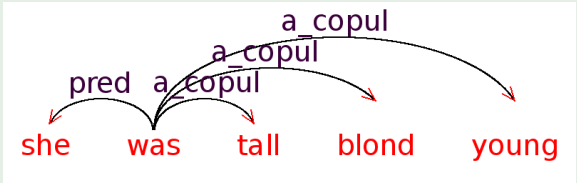
Idea of the Structural Bootstrapping Method (SBM)

SBM updates Extended CDG G_{in} for every input DS D

- if the vicinity $V(w, D_x)$ of a word w in D is generated by no RTE assigned to w in G_{in}
- then SBM finds a minimal more general Extended CDG G_{out}, i.e. $G_{in} \preceq G_{out}$, which generates $V(w, D)$

Vicinity Example : $V(was, D)$





On a *representative* sample of input DS $D_1, D_2, \ldots$

SBM would converge to an Extended CDG G generating the sample : $D_i \in \Delta(G), i = 1, 2, \ldots$

SBM applied to a DS D_x of x and Extended CDG $G_{in} = (W, \lambda)$, $W = \bigcup_{i \in I} C_i$

if $(D_x \in \Delta(G_{in}))$ then $G_{out} = G_{in}$
else
  for every word $w \in x$
    if $(w \in W)$
    then select a class C such that $w \in C$;
    else select a class C and add w to C
    end ;
    find the vicinity $V(w, D_x)$;
    if $(V(w, D_x)$ is generated by a RTE $t \in \lambda(C))$
    then $\lambda'(C) = \lambda(C)$
    else select RTE $t \in \lambda(C)$;
      find *minimal* RTE $t' \succ t$ *generating* $V(w, D_x)$;
      set $\lambda'(C) = (\lambda(C) - \{t\}) \cup \{t'\}$,
      $\lambda'(C_1) = \lambda(C_1)$ for every $C_1 \neq C$
    end
  until $D_x \in \Delta((W, \lambda'))$
end ;
return $G_{out} = (W, \lambda')$

Application of SBM. An Example :

Evolution of the class *Vt* of French transitive verbs

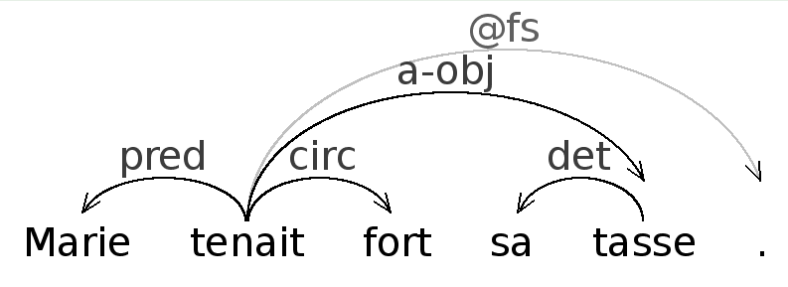
This is how SBM may update *Vt* when applied to DS of three sentences with transitive main verbs :
Marie tenait fort sa tasse (*Mary held tight her cup*)
Demain tu toucheras aussi ta paie (*Tomorrow you will get your wage too*)
Où mettrait-elle la clé ? (*Where might she put the key ?*)

Application of SBM. An Example :

Evolution of the class *Vt* of French transitive verbs

This is how SBM may update *Vt* when applied to DS of three sentences with transitive main verbs :
Marie tenait fort sa tasse (*Mary held tight her cup*)
Demain tu toucheras aussi ta paie (*Tomorrow you will get your wage too*)
Où mettrait-elle la clé ? (*Where might she put the key ?*)

EX1 : *Mary held tight her cup (Fr.)*



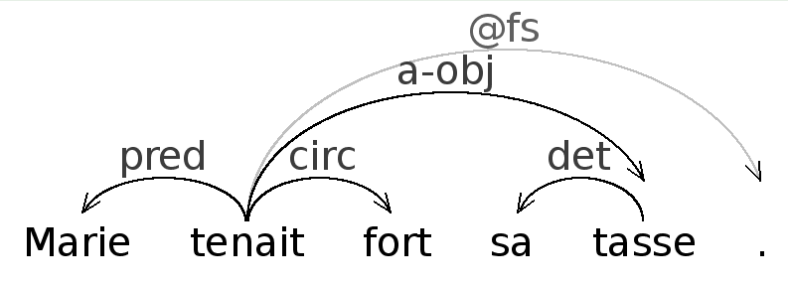
tenait (*held*)
tenait $\in$ *Vt*

Application of SBM. An Example :

Evolution of the class *Vt* of French transitive verbs

This is how SBM may update *Vt* when applied to DS of three sentences with transitive main verbs :
Marie tenait fort sa tasse (*Mary held tight her cup*)
Demain tu toucheras aussi ta paie (*Tomorrow you will get your wage too*)
Où mettrait-elle la clé ? (*Where might she put the key ?*)

EX1 : *Mary held tight her cup (Fr.)*

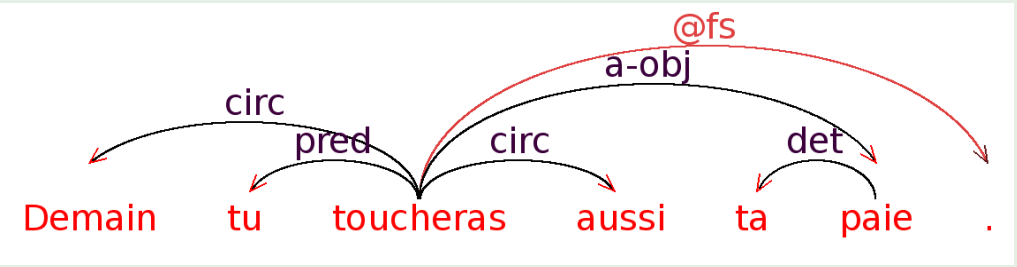


tenait (*held*)
tenait $\in$ *Vt*

$Vt \mapsto [pred \backslash S / @fs / a-obj / circ]$

Second Step

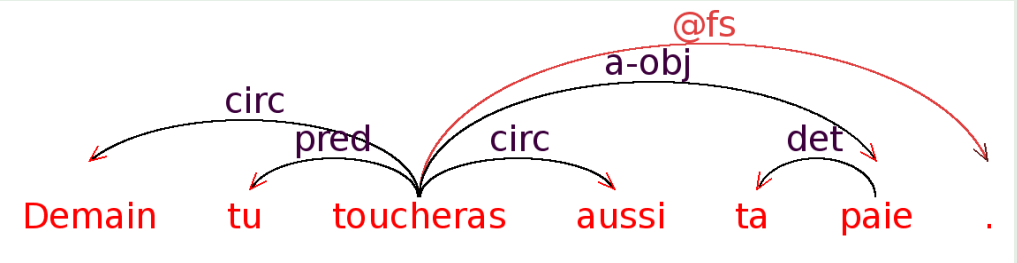
EX2 : *Tomorrow you will get your wage too (Fr.)*



toucheras
(*will get*)
toucheras $\in$ *Vt*

Second Step

EX2 : *Tomorrow you will get your wage too (Fr.)*



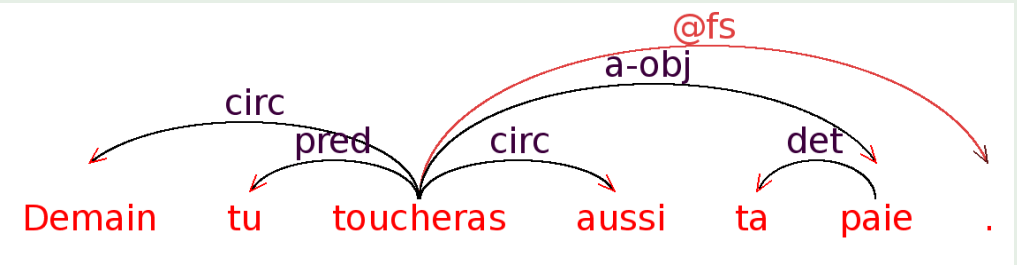
toucheras
(*will get*)
toucheras $\in$ *Vt*

a new *Vt* context :

- circumstantials in pre- and post-verb position
- repetitive circumstantial dependency (K=2)

Second Step

EX2 : *Tomorrow you will get your wage too (Fr.)*



toucheras
(*will get*)
toucheras $\in Vt$

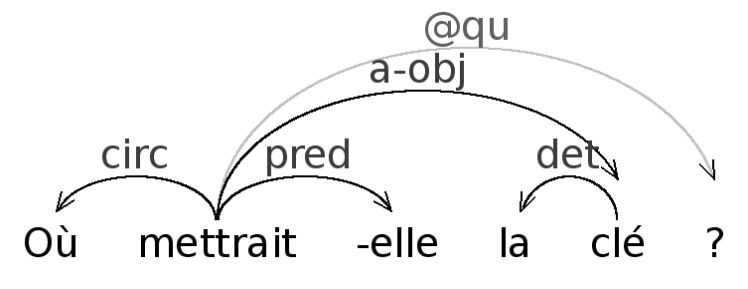
a new *Vt* context :

- circumstantials in pre- and post-verb position
- repetitive circumstantial dependency (K=2)

$$Vt \mapsto \{circ^*\}[pred\backslash S/@fs/a-obj]$$

Third Step

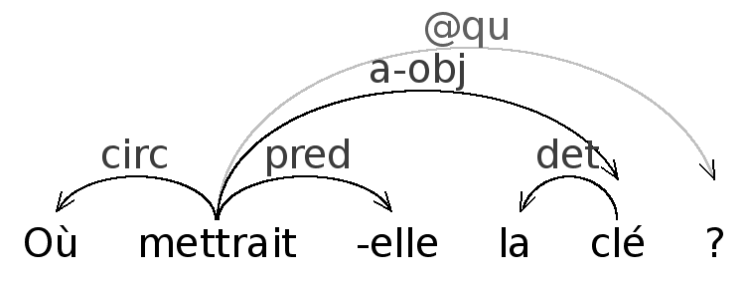
EX3 : *Where might she put the key ? (Fr.)*



mettrait (*might put*)
mettrait $\in$ *Vt*

Third Step

EX3 : *Where might she put the key ? (Fr.)*



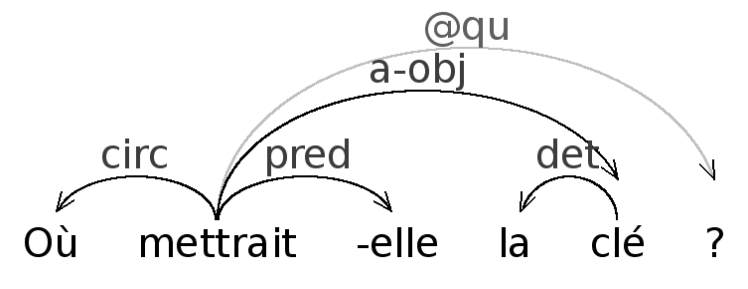
mettrait (*might put*)
mettrait $\in$ *Vt*

a new *Vt* context :

- question marker
- subject in post-verb position

Third Step

EX3 : *Where might she put the key ? (Fr.)*



mettrait (*might put*)
mettrait $\in Vt$

a new *Vt* context :

- question marker
- subject in post-verb position

$$Vt \mapsto \{pred, circ^*\}[S/(@fs|@qu)/a-obj]$$

Bootstrapping of Wide Coverage CDG

Application of SBM in practice :

- SBM is limited to the lexicon of the input sample ⇒ the developed grammar should be integrated with a morpho-syntactic dictionary
- needs an efficient parser supporting the search of adequate DS
- SBM is fully monotone, whereas in practice, the large scale grammars undergo non-monotone updates ⇒ it needs checking of correctnes of updated grammars wrt the input samples of DS

Bootstrapping of Wide Coverage CDG

Application of SBM in practice :

- SBM is limited to the lexicon of the input sample ⇒ the developed grammar should be integrated with a morpho-syntactic dictionary
- needs an efficient parser supporting the search of adequate DS
- SBM is fully monotone, whereas in practice, the large scale grammars undergo non-monotone updates ⇒ it needs checking of correctnes of updated grammars wrt the input samples of DS

Toolkit "CDGLab" [Alfared,Bechet,Dikovsky'Depling-2011]

- supports SBM with these means
- serves for parallel development of large scale CDG and of grammatically correct DS corpora

Case of a Wide Coverage CDG of French

Extended CDG of French (CDGF-3.2)

- kernel part bootstrapped from $\sim$ 400 DS
- integrated with Lefff 3.0 [Sagot'2010] (MS-Dictionary of French, 536,375 forms)
- current version CDGF-3.2 bootsrapped from $\sim$ 200 more DS (undergoed three global non-monotone revisions using the toolkit CDGLab)
- CDGF-3.2 covers the major part of French syntax and is provably correct wrt a corpus of $\sim$ 1600 DS

Case of a Wide Coverage CDG of French

Extended CDG of French (CDGF-3.2)

- kernel part bootstrapped from $\sim$ 400 DS
- integrated with Lefff 3.0 [Sagot'2010] (MS-Dictionary of French, 536,375 forms)
- current version CDGF-3.2 bootsrapped from $\sim$ 200 more DS (undergoed three global non-monotone revisions using the toolkit CDGLab)
- CDGF-3.2 covers the major part of French syntax and is provably correct wrt a corpus of $\sim$ 1600 DS

Grammar Complexity

- Sample size : $\sim$ 600 DS
- Granularity :185 lexical classes (46 verbal, 7 nominal)
- Grammar size : $\sim$ 3120 RTE
- Dependencies : 180 projective (= 84 parametrized) and 36 non-projective (= 20 parametrized) in 40 groups

Experimental DS corpus

Gold Standard DS corpus (GS)

- Actual size : 1630 DS
- Correct wrt CDGF-3.2 : $GS \subset \Delta(CDGF-3.2)$
- Non-projectve part : 42.8% of DS (n-p dependencies : negative, reflexive, clitic, co-referential, comparative, etc.)
- Formats : XML, DB, graphical (svg,png)

Lex Unit	Class	+ / 0 / -	Form	Lemma	Mood	Tense	Person	Gender	Number
Elle	PN(Lex=pers,C=n)	◉ ○ ○	elle	il			3	f	s
	N(Lex=proper)	○ ○ ◉	Elle	Elle				f	s
la	PN(Lex=pn,F=clit,C=a)	◉ ○ ○	la	le			3	f	s
lui	PN(Lex=pn,F=clit,P=3,C=d)	◉ ○ ○	lui	lui			3		s
a	Vaux(Lex=avoir,F=fin)	◉ ○ ○	a	avoir	indicatif	présent	3		s
donnée	V2t(F=pz,C1=a,C2=d|g|l,T=past)	◉ ○ ○	donnée	donner	participe	passé		f	s
.	FullStop(Lex='.')	◉ ○ ○	.	.					

Conclusions

Extended CDG

- express unlimited DS and are well adapted to a practical development of wide coverage dependency grammars

Conclusions

Extended CDG

- express unlimited DS and are well adapted to a practical development of wide coverage dependency grammars
- express voluminous sets of types using well-structured and succinct RTE assigned to lexical classes

Conclusions

Extended CDG

- express unlimited DS and are well adapted to a practical development of wide coverage dependency grammars
- express voluminous sets of types using well-structured and succinct RTE assigned to lexical classes
- are analyzed in a reasonable polynomial time due to the extended type calculus

Conclusions

Extended CDG

- express unlimited DS and are well adapted to a practical development of wide coverage dependency grammars
- express voluminous sets of types using well-structured and succinct RTE assigned to lexical classes
- are analyzed in a reasonable polynomial time due to the extended type calculus

Sructural Bootstrapping Method supported by CDGLab allows

- to incrementally develop and update large scale dependency grammars in a short space of time

Conclusions

Extended CDG

- express unlimited DS and are well adapted to a practical development of wide coverage dependency grammars
- express voluminous sets of types using well-structured and succinct RTE assigned to lexical classes
- are analyzed in a reasonable polynomial time due to the extended type calculus

Sructural Bootstrapping Method supported by CDGLab allows

- to incrementally develop and update large scale dependency grammars in a short space of time
- to create and check high quality DS corpora provably correct wrt evolving CDG

Conclusions

Extended CDG

- express unlimited DS and are well adapted to a practical development of wide coverage dependency grammars
- express voluminous sets of types using well-structured and succinct RTE assigned to lexical classes
- are analyzed in a reasonable polynomial time due to the extended type calculus

Sructural Bootstrapping Method supported by CDGLab allows

- to incrementally develop and update large scale dependency grammars in a short space of time
- to create and check high quality DS corpora provably correct wrt evolving CDG

THANK YOU !

CDG Definition

CDG $G = (W, \mathbf{Cat}, S, \lambda)$:

Dictionary : W

Types : the set of (polarized) categories **Cat**, $S \in \mathbf{Cat}$ (axiom)

Lexicon (type assignment) : $\lambda : W \to 2^{\mathbf{Cat}}$ (i.e. $w \mapsto \{c_1, ..., c_k\} \subset \mathbf{Cat}$).

Calculus (one for all CDG) : $\alpha_1 ... \alpha_n \vdash \alpha, \quad \alpha, \alpha_1, ..., \alpha_n \in \mathbf{Cat}$

Language : $L(G) = \{x \in W^+ \mid (\exists\, \Gamma)\ (\Gamma \in \lambda(x)\ \wedge\ \Gamma \vdash^* S)\}$

DS-language : $D(G) =$
$\{D(\pi) \mid (\exists\, x)(\exists\, \Gamma)\ (x \in L(G)\ \wedge\ \Gamma \in \lambda(x)\ \wedge\ \pi = \Gamma \vdash^* S)\}$

EX : $L(G_{abc}) = \{a^n b^n c^n \mid n > 0\}$ for the CDG G_{abc} :
$a \mapsto A^{\swarrow A},\ [A \backslash A]^{\swarrow A},$
$b \mapsto [B/C]^{\nwarrow A},\ [A \backslash S/C]^{\nwarrow A},$
$c \mapsto C,\ [B \backslash C]$

Proofs of typing correctness

EX : *Elle la lui a donnée*

Without anchoring (clitics may permute) :

$[pred]\ [\varepsilon]^{\swarrow clit-dobj}\ [\varepsilon]^{\swarrow clit-iobj}\ [pred\backslash S/aux-a-d]\ [aux-a-d]^{\nwarrow clit-iobj,\nwarrow clit-dobj}$

$$\dfrac{\dfrac{[pred]\quad \dfrac{[\varepsilon]^{\swarrow clit-dobj}\quad \dfrac{[\varepsilon]^{\swarrow clit-iobj}[pred\backslash S/aux]}{[pred\backslash S/aux]^{\swarrow clit-iobj}}(\mathbf{L}^l)}{[pred\backslash S/aux]^{\swarrow clit-dobj\swarrow clit-iobj}}(\mathbf{L}^l)}{[S/aux]^{\swarrow clit-dobj\swarrow clit-iobj}}(\mathbf{L}^l)\qquad [aux]^{\nwarrow clit-iobj\nwarrow clit-dobj}}{[S]^{\swarrow clit-dobj\swarrow clit-iobj\nwarrow clit-iobj\nwarrow clit-dobj}}(\mathbf{L}^r)}{S}(\mathbf{D}^l\times 2)$$

With anchoring (clitics cannot permute) :

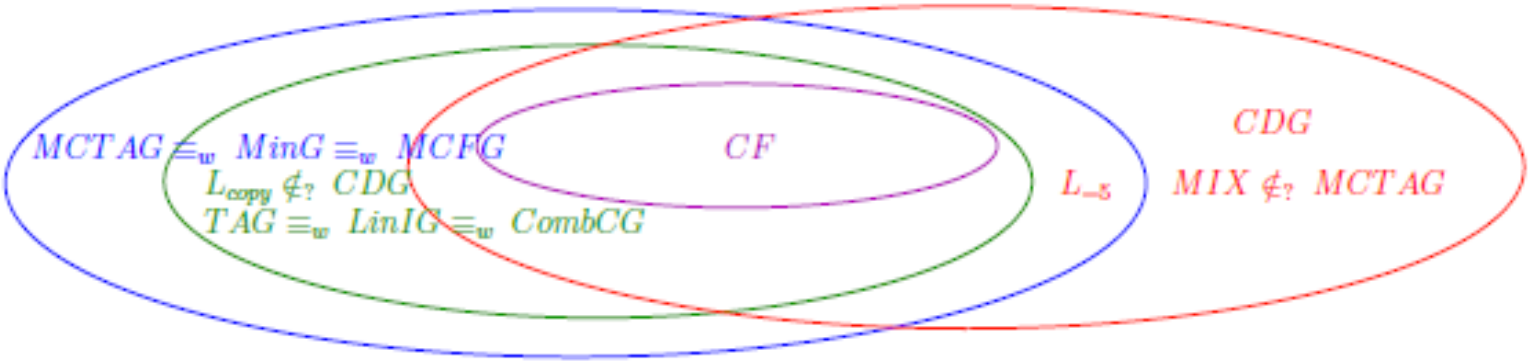
$[pred]\ [\#^l(\swarrow clit-dobj)]^{\swarrow clit-dobj}\ [\#^l(\swarrow clit-iobj)]^{\swarrow clit-iobj}\ [\#^l(\swarrow clit-iobj)\backslash\#^l(\swarrow clit-dobj)\backslash pred\backslash S/aux-a-d]\ [aux-a-d]^{\nwarrow clit-iobj,\nwarrow clit-dobj}$

$$\dfrac{\dfrac{[pred]\quad \dfrac{[\#^l(\swarrow clit-dobj)]^{\swarrow clit-dobj}\quad \dfrac{[\#^l(\swarrow clit-iobj)]^{\swarrow clit-iobj}[\#^l(\swarrow clit-iobj)\backslash\#^l(\swarrow clit-dobj)\backslash pred\backslash S/aux]}{[\#^l(\swarrow clit-dobj)\backslash pred\backslash S/aux]^{\swarrow clit-iobj}}(\mathbf{L}^l)}{[pred\backslash S/aux]^{\swarrow clit-dobj\swarrow clit-iobj}}(\mathbf{L}^l)}{[S/aux]^{\swarrow clit-dobj\swarrow clit-iobj}}(\mathbf{L}^l)\qquad [aux]^{\nwarrow clit-iobj\nwarrow clit-dobj}}{[S]^{\swarrow clit-dobj\swarrow clit-iobj\nwarrow clit-iobj\nwarrow clit-dobj}}(\mathbf{L}^r)}{S}(\mathbf{D}^l\times 2)$$

Properties of CDG

Parsing complexity (Dekhtyar,Dikovsky'2004,2008) :

1. theoretical : $O(n^{3+2p})$ (for p polarized valencies)
2. in practice: $O(n^4)$ ($O(n^3)$ for projective dependencies).



$L_{copy} = \{ww \mid w \in W^+\}$, $L_{=i} = \{a_1^n \dots a_i^n \mid n > 0\}$, $MIX = \{w \in \{a,b,c\}^+ \mid |w|_a = |w|_b = |w|_c\}$