On Dispersed and Choice Iteration in Incrementally Learnable Dependency Types

D. **Béchet**, A. **Dikovsky** (LINA), Nantes
and A. **Foret** (IRISA), Rennes, France

LACL'2011, Montpellier, June 29,30, July 1, 2011

Main points

- THESIS : surface syntactic dependencies between words are determined by words' valencies
- CONSEQUENCE : dependency grammars cannot be learned from dependency structures they generate
  because of repeatable dependencies

Main points

- THESIS : surface syntactic dependencies between words are determined by words' valencies
- CONSEQUENCE : dependency grammars cannot be learned from dependency structures they generate
  because of repeatable dependencies
- MAIN RESULT : grammars treating the repeatable dependencies in a linguistically proper way are

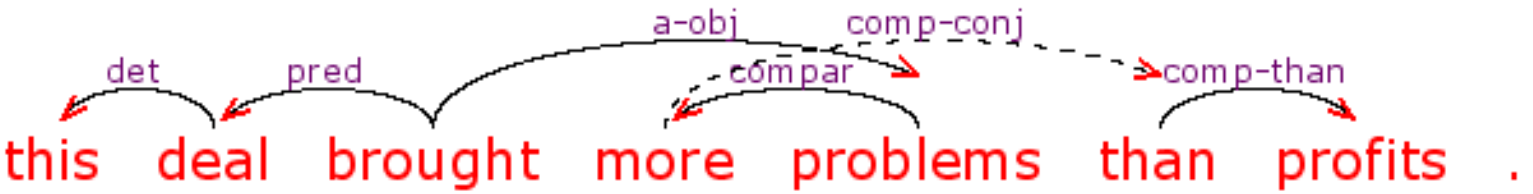
  incrementally learnable from dependency structures

Plan

Basics of Dependency Syntax

Surface Dependency Structures (DS) are graphs of surface syntactic relations between the *words* in a sentence.

A Dependency Structure



Dependencies are determined by valencies of words

brought has +valency pred of a left adjacent word

deal has −valency pred of a right adjacent word

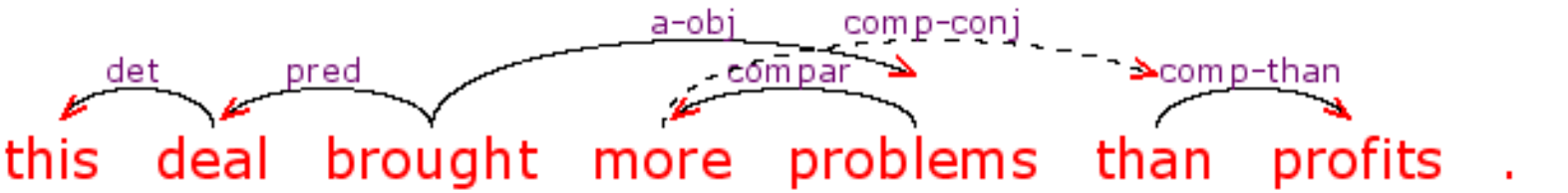
Saturation of valency pred determines projective dependency

brought $\overset{pred}{\longleftarrow}$ *deal* (**Governor :** brought, **Subordinate :** deal)

Basics of Dependency Syntax

Surface Dependency Structures (DS) are graphs of surface syntactic relations between the *words* in a sentence.

A Dependency Structure



Dependencies are determined by valencies of words

more has +valency comp-conj of a word somewhere on its right

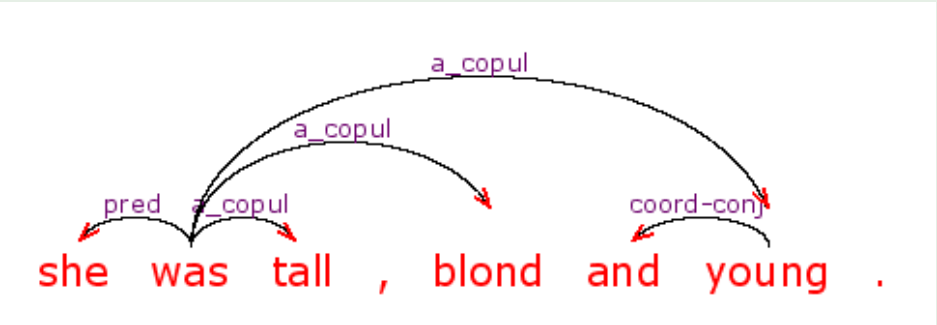
than has −valency comp-conj of a word somewhere on its left

Saturation of comp-conj determines discontinuous dependency

more $\overset{comp-conj}{-->}$ *than* (**Governor :** more, **Subordinate :** than)

Repeatable dependencies

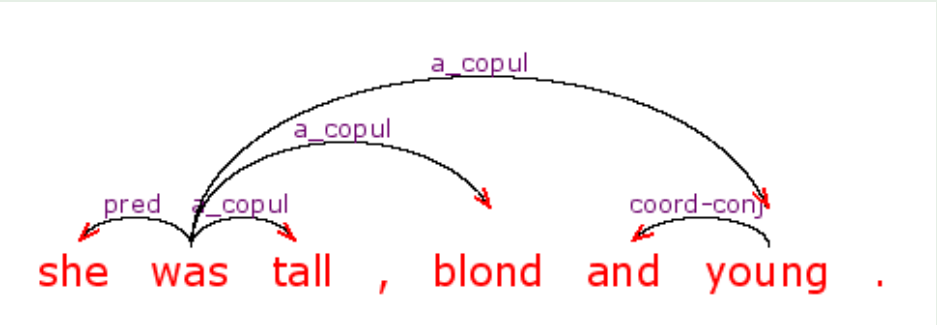
Some dependency valencies are MULTIPLE



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

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies

PROJECTIVE DEPENDENCIES

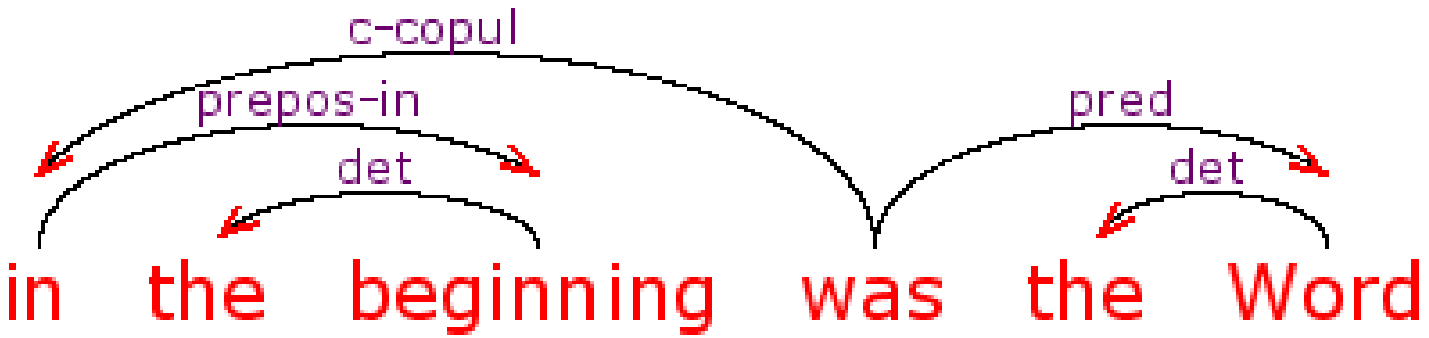
Dependency : $Gov \xrightarrow{d} Sub$:

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

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

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies



$in \mapsto [c\_copul / prepos - in]$
$the \mapsto [det]$
$beginning \mapsto [det \backslash prepos - in]$
$was \mapsto [c\_copul \backslash S / pred]$
$Word \mapsto [det \backslash pred]$

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies

NON-PROJECTIVE DEPENDENCIES

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

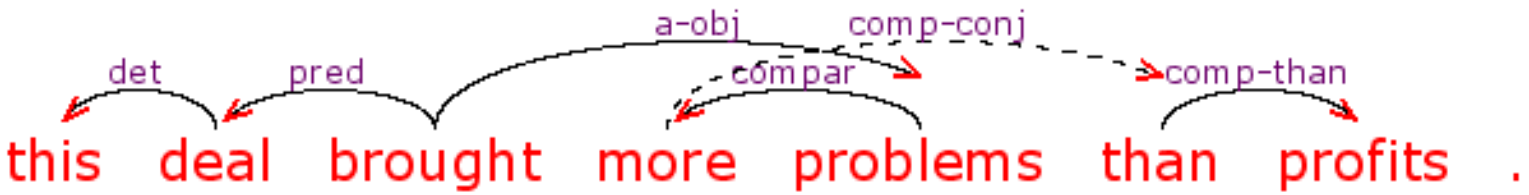
Dependency : $Gov \overset{d}{-\!\!-\!\!>} Sub$:

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

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

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies



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

CDG calculus

Left-oriented rules

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

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. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$ $\quad Gov \xrightarrow{C} Sub$

I^l. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1P_2}$ $\quad Gov \xrightarrow{C} Sub$

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

D^l. $\alpha^{P_1}(\swarrow C)P(\nwarrow C)^{P_2} \vdash \alpha^{P_1}PP_2$ $\quad Gov \overset{C}{-\!-\!>} Sub$

First-Available Rule

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 $\swarrow C$, $\nwarrow C$

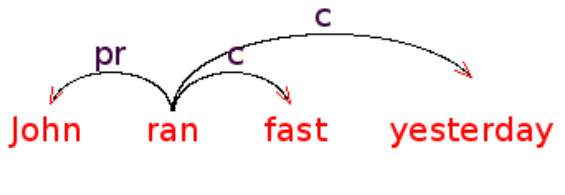
LEXICON :
(empty potentials)

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

Derivation

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

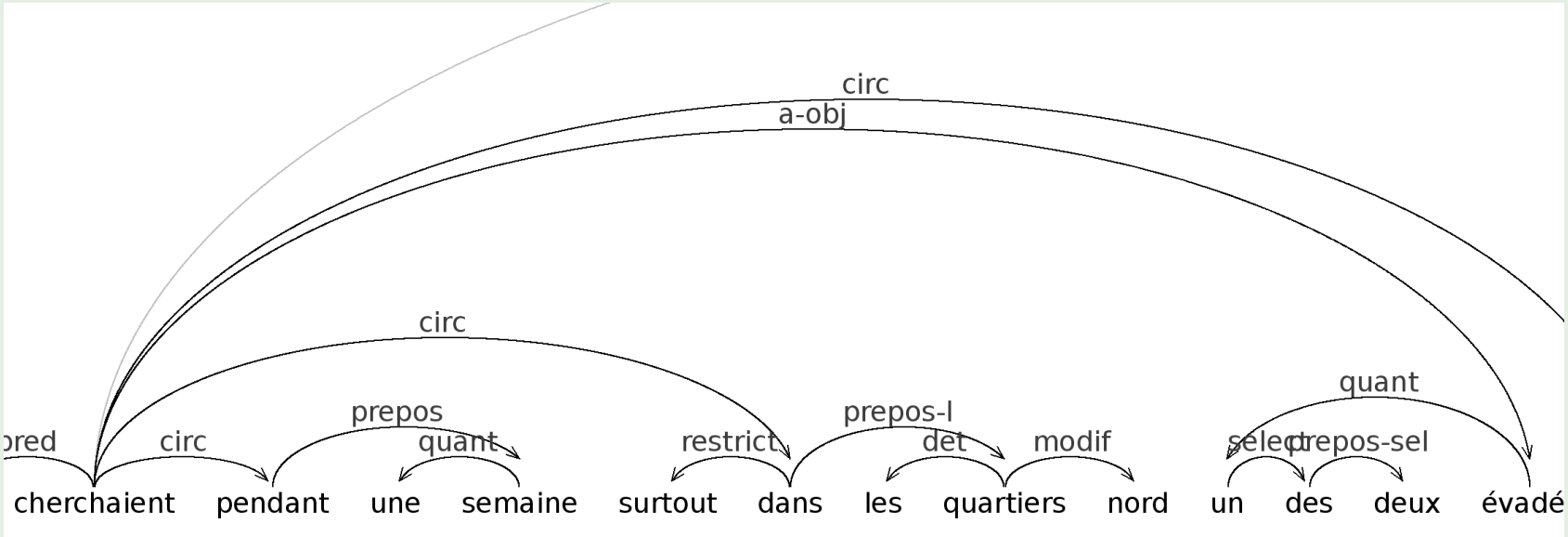
Dependency structures



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

Interpretations of Repeatable Dependencies

Which dependencies are repeatable ?



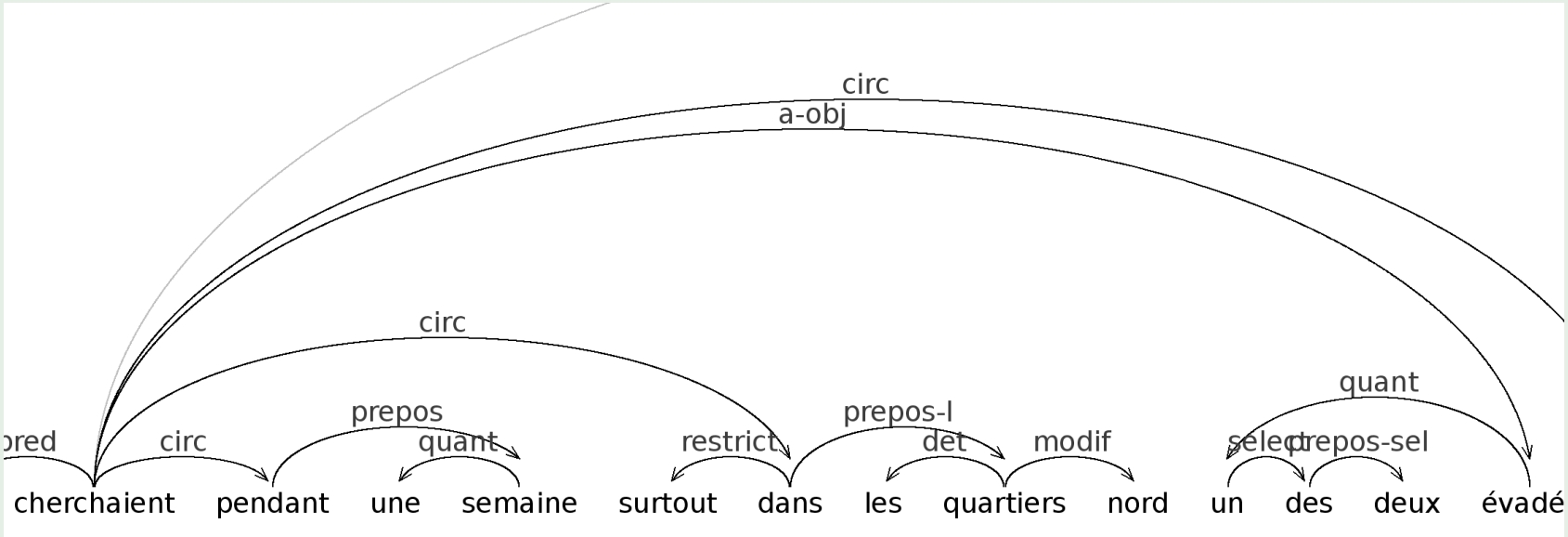
"They tracked for a week especially in the nord quarters one of the two fugitives systematically blocking entries and exits"

Word Order and Repeatability

Is the dependency circ repeatable for $K = 3$?

Interpretations of Repeatable Dependencies

Which dependencies are repeatable ?



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

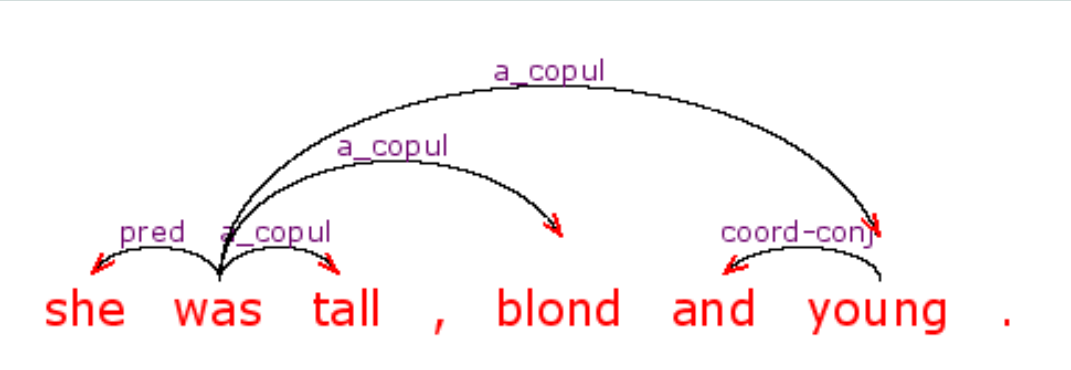
Word Order and Repeatability

PB : Repeatable Dependency Principle is uncertain about the WO

Three Readings of Repeatable Dependencies

Sequentially Repeatable Dependencies and Iterated Types

EX : sequentially repeatable copula dependency

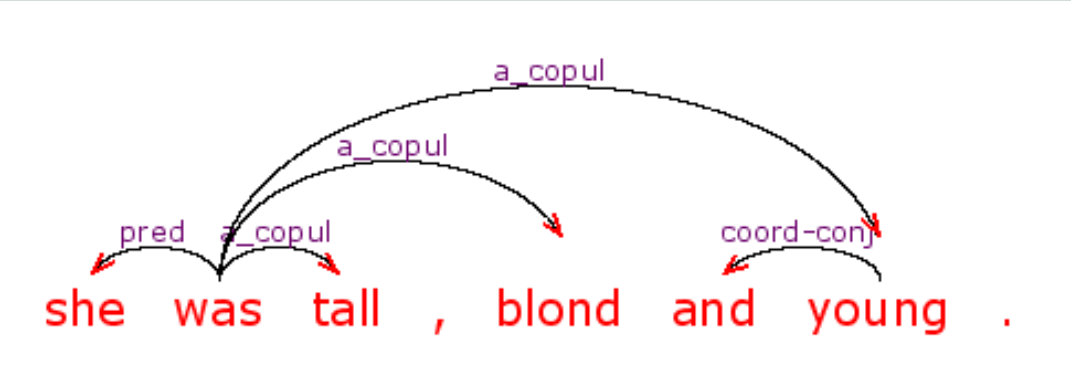


$was \mapsto [pred \backslash S / a\_copul^*]$

Three Readings of Repeatable Dependencies

Sequentially Repeatable Dependencies and Iterated Types

EX : sequentially repeatable copula dependency



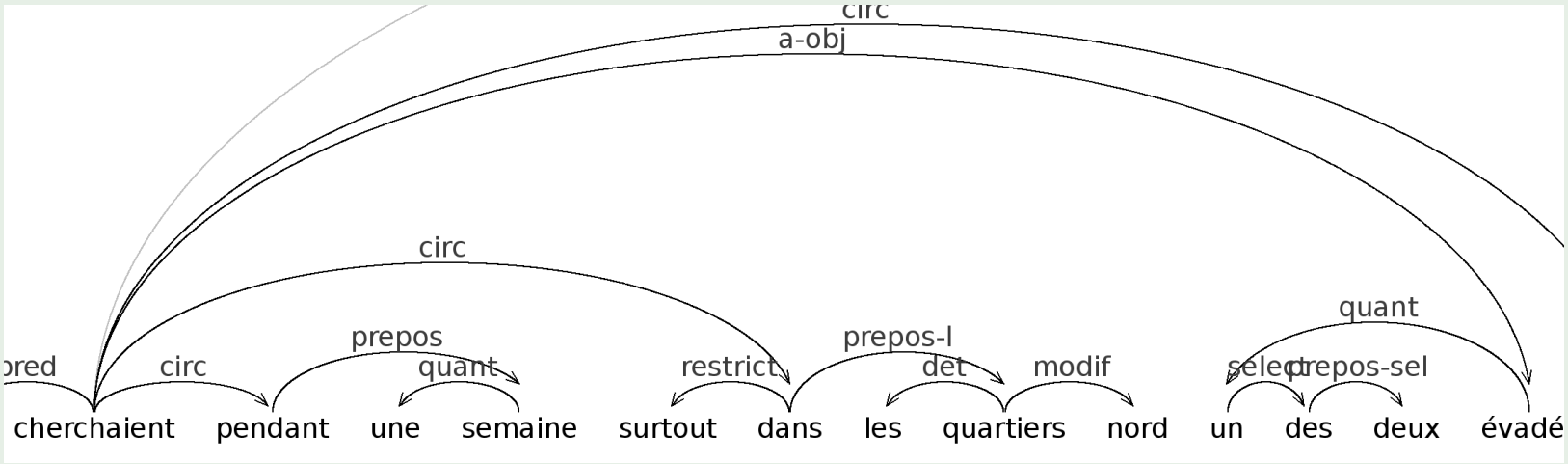
$was \mapsto [pred \backslash S / a\_copul^*]$

Iterated Types of CDG

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

Dispersed Iteration Types

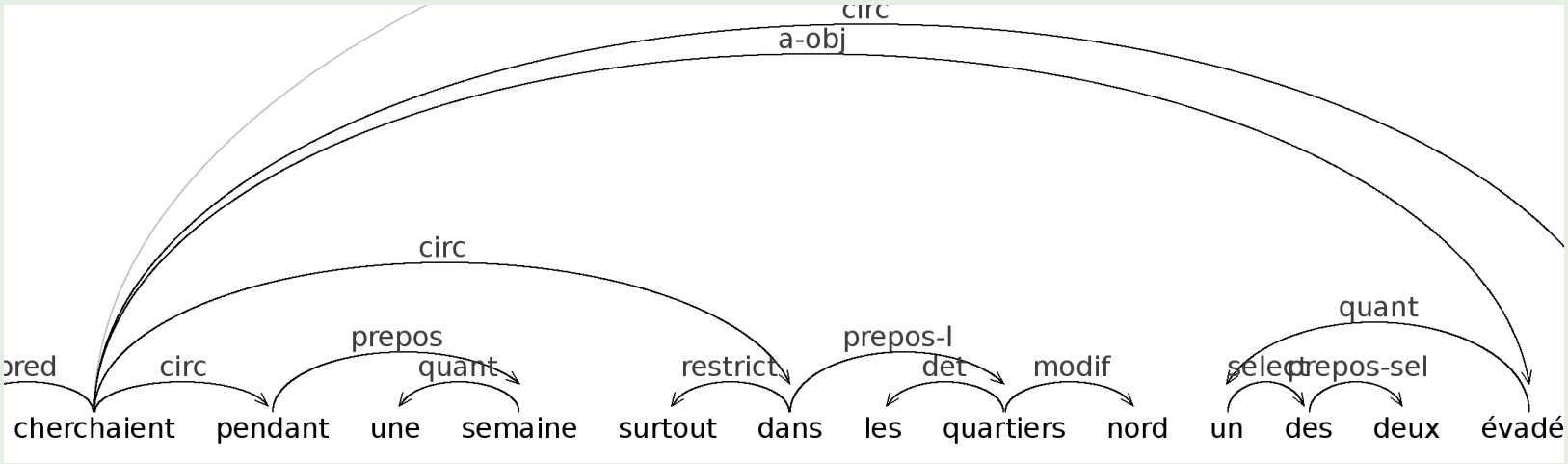
Dispersed Repeatable Dependencies ($K = 3$)



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

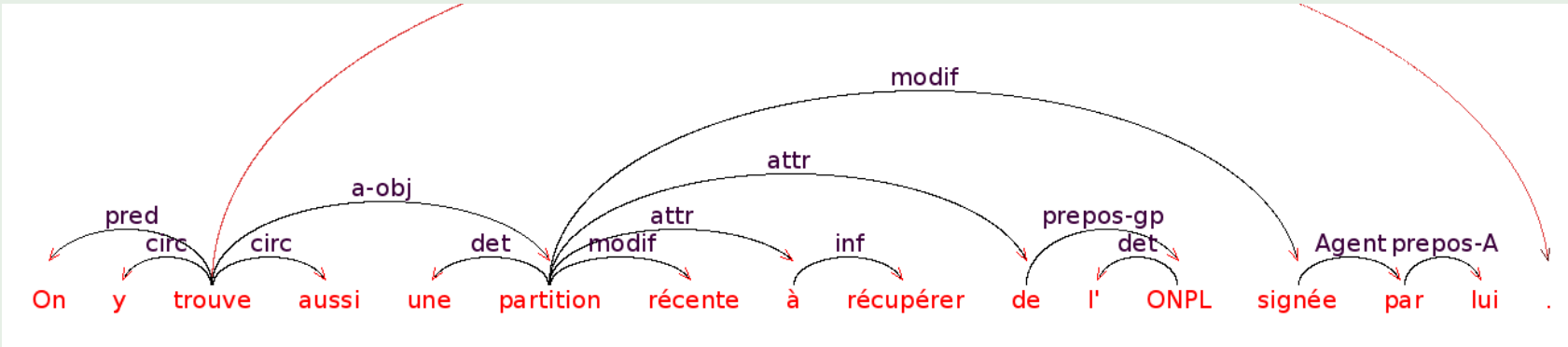
$$\mathbf{LD^I}.\quad C^{P_1}[\{\alpha_1\}\backslash C\backslash\beta/\{\alpha_2\}]^{P_2} \vdash [\{\alpha_1\}\backslash\beta/\{\alpha_2\}]^{P_1P_2}$$

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

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

Choice Iteration Types

Choice Repeatable Dependencies ($K = 2$)

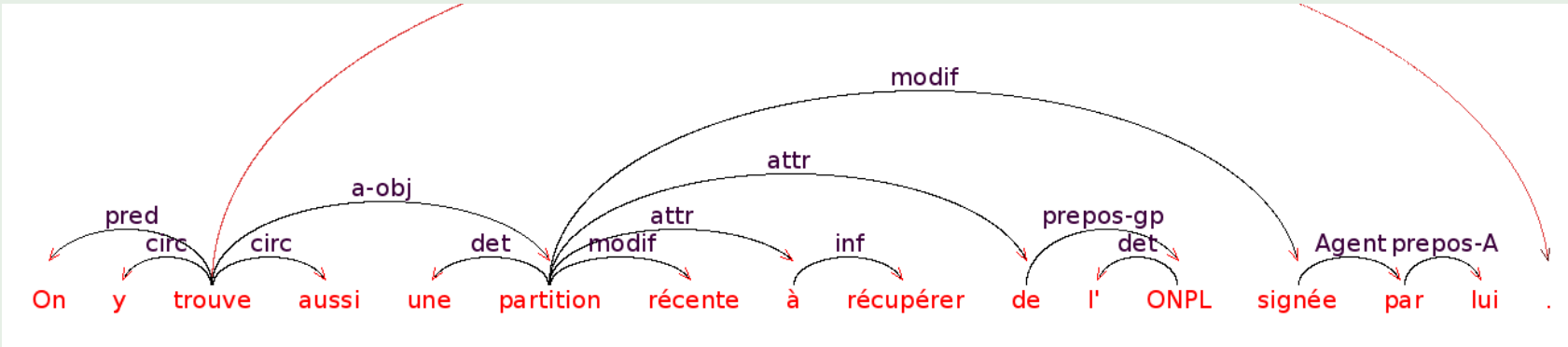


*One here finds also a partition recent to obtain of ONPL signed by him

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

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

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

Grammatical inference

symbolic, from positive examples [Gold'67]

Grammatical class $\mathcal{G}$ is learnable if there is an algorithm A which

- for every target grammar $G_T \in \mathcal{G}$
- every enumeration $\sigma = L(G_T)$ and every prefix $\sigma[n]$,

returns a hypothetical grammar $A(\sigma[n]) \in \mathcal{G}$ **and** :

(i) the sequence of languages $\{L(A(\sigma[n])) \mid n \in \mathbb{N}\}$ converges to the target language $L(G_T)$

(ii) this holds for all enumerations σ of $L(G_T)$

Grammatical inference

symbolic, from positive examples [Gold'67]

Grammatical class $\mathcal{G}$ is learnable if there is an algorithm A which

- for every target grammar $G_T \in \mathcal{G}$
- every enumeration $\sigma = L(G_T)$ and every prefix $\sigma[n]$,

returns a hypothetical grammar $A(\sigma[n]) \in \mathcal{G}$ **and** :

(i) the sequence of languages $\{L(A(\sigma[n])) \mid n \in \mathbb{N}\}$ converges to the target language $L(G_T)$

(ii) this holds for all enumerations σ of $L(G_T)$

Learning from **strings** : $\sigma(\mathbb{N}) = L(G_T)$ / **structures** : $\sigma(\mathbb{N}) = \Delta(G_T)$

Grammatical inference

symbolic, from positive examples [Gold'67]

Grammatical class $\mathcal{G}$ is learnable if there is an algorithm A which

- for every target grammar $G_T \in \mathcal{G}$
- every enumeration $\sigma = L(G_T)$ and every prefix $\sigma[n]$,

returns a hypothetical grammar $A(\sigma[n]) \in \mathcal{G}$ **and** :

(i) the sequence of languages $\{L(A(\sigma[n])) \mid n \in \mathbb{N}\}$ converges to the target language $L(G_T)$

(ii) this holds for all enumerations σ of $L(G_T)$

Learning from **strings** : $\sigma(\mathbb{N}) = L(G_T)$ / **structures** : $\sigma(\mathbb{N}) = \Delta(G_T)$

Limit points and non-learnability

A limit point in a family $\mathcal{L}$ is a sequence $< L_n >_{n\in\mathbb{N}} \in \mathcal{L}$ such that :

$$\forall i : L_i \subsetneq L_{i+1} \text{ and } \bigcup_{n\in\mathbb{N}} L_n \in \mathcal{L}.$$

Limit Points entail the non-learnability of $\mathcal{L}$.

Unlearnability of Unlimited CDG

Unlearnability of CDG from strings (FG'2004)

TH k-rigid CDG without iterated types are learnable from FA-like structures (and from strings).

TH (1)-rigid CDG are not learnable from strings (a limit point).

Unlearnability of Unlimited CDG

Unlearnability of CDG from strings (FG'2004)

TH k-rigid CDG without iterated types are learnable from FA-like structures (and from strings).

TH (1)-rigid CDG are not learnable from strings (a limit point).

Limit point

$$G'_0 = \{a \mapsto A, b \mapsto B, c \mapsto C'_0\} \qquad C'_0 = S$$
$$G'_n = \{a \mapsto A, b \mapsto B, c \mapsto [C'_n]\} \qquad C'_{n+1} = C'_n \,/\, \mathbf{A}^* \,/\, \mathbf{B}^*$$
$$G'_* = \{a \mapsto A, b \mapsto A, c \mapsto [S \,/\, \mathbf{A}^*]\}$$

$L(G'_n) = \{c(b^*a^*)^k \mid k \leq n\}$ and $L(G'_*) = c\{b, a\}^*$.

Unlearnability of Unlimited CDG

Unlearnability of CDG from strings (FG'2004)

TH k-rigid CDG without iterated types are learnable from FA-like structures (and from strings).

TH (1)-rigid CDG are not learnable from strings (a limit point).

Limit point

$$G'_0 = \{a \mapsto A, b \mapsto B, c \mapsto C'_0\} \qquad C'_0 = S$$
$$G'_n = \{a \mapsto A, b \mapsto B, c \mapsto [C'_n]\} \qquad C'_{n+1} = C'_n \,/\, \mathbf{A}^* \,/\, \mathbf{B}^*$$
$$G'_* = \{a \mapsto A, b \mapsto A, c \mapsto [S \,/\, \mathbf{A}^*]\}$$

$L(G'_n) = \{c(b^*a^*)^k \mid k \leq n\}$ and $L(G'_*) = c\{b, a\}^*$.

TH (1)-rigid CDG are not learnable from DS (FG'2010).
So the CDG are not learnable from dependency treebanks !

Incremental Learning (e.g. of Dispersed Iteration CDG)

CDG complying with the Principle of Repeatable Dependencies are incrementally learnable from DS.

Incremental Learning (e.g. of Dispersed Iteration CDG)

CDG complying with the Principle of Repeatable Dependencies are incrementally learnable from DS.

Incremental Learning from DS in $\Delta(G)$

Generalization PO $\preceq$: a PO on CDG compatible with the inclusion of DS-languages : $G \preceq G' \Rightarrow \Delta(G) \subseteq \Delta(G')$

Incremental Learning (e.g. of Dispersed Iteration CDG)

CDG complying with the Principle of Repeatable Dependencies are incrementally learnable from DS.

Incremental Learning from DS in $\Delta(G)$

Generalization PO $\preceq$: a PO on CDG compatible with the inclusion of DS-languages : $G \preceq G' \Rightarrow \Delta(G) \subseteq \Delta(G')$

Inference Algorithm $\mathcal{A}$ for CDG from DS is INCREMENTAL

if for every training sequence (enumeration) σ of $\Delta(G)$ $\mathcal{A}$ is :

1. `monotonic` on σ, i.e. $\mathcal{A}(\sigma[i]) \preceq \mathcal{A}(\sigma[j])$ for all $i \leq j$.
2. `faithful` on σ, i.e. $\Delta(\mathcal{A}(\sigma[i])) \subseteq \Delta(G)$ for all i.
3. `expansive` on σ, i.e. $\sigma[i] \subseteq \Delta(\mathcal{A}(\sigma[i]))$ for all i.

Incremental Learning (e.g. of Dispersed Iteration CDG)

CDG complying with the Principle of Repeatable Dependencies are incrementally learnable from DS.

Incremental Learning from DS in $\Delta(G)$

Generalization PO $\preceq$: a PO on CDG compatible with the inclusion of DS-languages : $G \preceq G' \Rightarrow \Delta(G) \subseteq \Delta(G')$

Inference Algorithm $\mathcal{A}$ for CDG from DS is INCREMENTAL

if for every training sequence (enumeration) σ of $\Delta(G)$ $\mathcal{A}$ is :

1. `monotonic` on σ, i.e. $\mathcal{A}(\sigma[i]) \preceq \mathcal{A}(\sigma[j])$ for all $i \leq j$.
2. `faithful` on σ, i.e. $\Delta(\mathcal{A}(\sigma[i])) \subseteq \Delta(G)$ for all i.
3. `expansive` on σ, i.e. $\sigma[i] \subseteq \Delta(\mathcal{A}(\sigma[i]))$ for all i.

TH If $\mathcal{A}$ is monotonic, faithful, expansive and stabilizes on σ then $\lim_{i \to \infty} \mathcal{A}(\sigma[i]) \equiv_s G$ $\quad (G \equiv_s G'$ means $\Delta(G) = \Delta(G'))$.

CDG Complying with RD Principle $\sim$ K-Star Revealing

Left (Right) Dispersed K-star Generalization

Of TYPE $t = [\{\alpha\}\backslash\beta_1 d\backslash \ldots \beta_n d\backslash\beta]^P$, $n \geq K$ are TYPES :

$$t_L = [\{\alpha, d^*\}\backslash\beta_1 \ldots \beta_n\beta]^P \text{ and } t_\Omega = [\{\alpha\}\backslash\beta_1 \ldots \beta_n\beta]^P$$

CDG Complying with RD Principle $\sim$ K-Star Revealing

Left (Right) Dispersed K-star Generalization

Of TYPE $t = [\{\alpha\}\backslash\beta_1 d\backslash \ldots \beta_n d\backslash\beta]^P$, $n \geq K$ are TYPES :

$$t_L = [\{\alpha, d^*\}\backslash\beta_1 \ldots \beta_n\beta]^P \text{ and } t_\Omega = [\{\alpha\}\backslash\beta_1 \ldots \beta_n\beta]^P$$

Of CDG : recursively

$$G = (G' \cup \{w \mapsto t\}) \Rightarrow (G \cup \{w \mapsto t_L, w \mapsto t_\Omega\})$$

till stabilization on $\mathcal{C}^K_{disp}(G)$: K-star Generalization of G

CDG Complying with RD Principle $\sim$ K-Star Revealing

Left (Right) Dispersed K-star Generalization

Of TYPE $t = [\{\alpha\}\backslash\beta_1 d\backslash \ldots \beta_n d\backslash\beta]^P$, $n \geq K$ are TYPES :

$$t_L = [\{\alpha, d^*\}\backslash\beta_1 \ldots \beta_n\beta]^P \text{ and } t_\Omega = [\{\alpha\}\backslash\beta_1 \ldots \beta_n\beta]^P$$

Of CDG : recursively

$$G = (G' \cup \{w \mapsto t\}) \Rightarrow (G \cup \{w \mapsto t_L, w \mapsto t_\Omega\})$$

till stabilization on $\mathcal{C}^K_{disp}(G)$: K-star Generalization of G

Dispersed K-star Revealing CDG

CDG G is dispersed K-star revealing for $K > 1$ if $G \equiv_s \mathcal{C}^K_{disp}(G)$

CDG Complying with RD Principle $\sim$ K-Star Revealing

Left (Right) Dispersed K-star Generalization

Of TYPE $t = [\{\alpha\}\backslash\beta_1 d\backslash \ldots \beta_n d\backslash\beta]^P$, $n \geq K$ are TYPES :

$$t_L = [\{\alpha, d^*\}\backslash\beta_1 \ldots \beta_n\beta]^P \text{ and } t_\Omega = [\{\alpha\}\backslash\beta_1 \ldots \beta_n\beta]^P$$

Of CDG : recursively

$$G = (G' \cup \{w \mapsto t\}) \Rightarrow (G \cup \{w \mapsto t_L, w \mapsto t_\Omega\})$$

till stabilization on $\mathcal{C}^K_{disp}(G)$: K-star Generalization of G

Dispersed K-star Revealing CDG

CDG G is dispersed K-star revealing for $K > 1$ if $G \equiv_s \mathcal{C}^K_{disp}(G)$

For $K = 2$ and $G(t) : x \mapsto [a]$, $y \mapsto [b]$, $z \mapsto t$

$G([\{a^*\}\backslash b\backslash S/a/b])$ and $G([a\backslash b\backslash S/\{a, b\}])$ are 2-star revealing
$G([a\backslash a\backslash S])$ and $G([a\backslash S/b/a/b])$ are not.

Learning Algorithm for Dispersed K-star revealing CDG

- is INCREMENTAL wrt the genericity PO $<_{disp}$ induced by :

$[\{\alpha\}\backslash\beta_1 d\backslash\ldots\beta_n d\backslash\beta]^P < [\{\alpha, d^*\}\backslash\beta_1\ldots\beta_n\beta]^P$, $n \geq K$
$[\{\alpha\}\backslash\beta]^P < [\{\alpha, d^*\}\backslash\beta]^P$

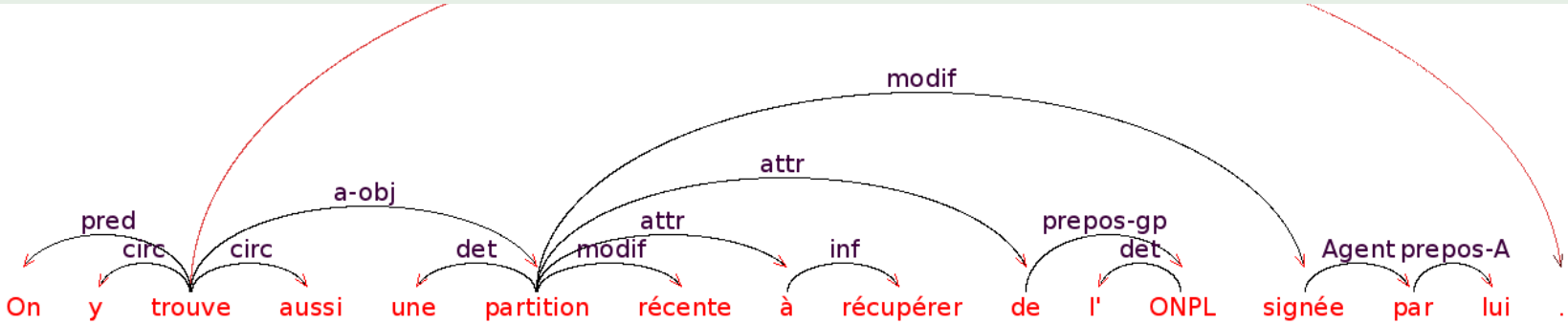
Learning Algorithm for Dispersed K-star revealing CDG

- is INCREMENTAL wrt the genericity PO $<_{disp}$ induced by :

$[\{\alpha\}\backslash\beta_1 d\backslash\ldots\beta_n d\backslash\beta]^P < [\{\alpha, d^*\}\backslash\beta_1\ldots\beta_n\beta]^P,\ n \geq K$
$[\{\alpha\}\backslash\beta]^P < [\{\alpha, d^*\}\backslash\beta]^P$

- computes words' types from their VICINITIES in DS

Vicinity $V(w, D)$ of word w in DS D :



$V(partition, D) = [det\backslash a-obj/modif/attr/attr/modif]$,
$V(de, D) = [attr/prepos-gp]$

Algorithm $\mathbf{TGE}^{(K)}_{disp}$ (type-generalize-expand) :
Input : $\sigma[i]$ (σ being a training sequence).
Output : GCD $\mathbf{TGE}^{(K)}_{disp}(\sigma[i])$.
let G_H be CDG with lexicon λ_H
$\lambda_H := \emptyset$;
(loop) **for** $i \geq 0$ //Infinite loop on σ
let $\sigma[i+1] = \sigma[i] \cdot D$;
let $(x, E) = D$; // D is the next example
(loop) **for every** $w \in x$ // w is a word in D
let $V(w, D) = [l_m \backslash \cdots \backslash l_1 \backslash h / r_1 / \cdots / r_n]^P$ // vicinity of w in D
let $LT := \{l_1\} \cup \cdots \cup \{l_m\}$
let $LF := \{d : d \in LT, card(\{i : 1 \leq i \leq m, l_i = d\}) \geq K\}$ // left K-repeatable
let $RT := \{r_1\} \cup \cdots \cup \{r_n\}$
let $RF := \{d : d \in RT, card(\{i : 1 \leq i \leq n, r_i = d\}) \geq K\}$ // right K-repeatable
let $t_w := [\{lf_1^*, \ldots, lf_p^*\} \backslash l'_{m'} \backslash \cdots \backslash l'_1 \backslash h / r'_1 / \cdots / r'_{n'} / \{rf_1^*, \ldots, rf_q^*\}]^P$ // generalised type
where $\{lf_1, \ldots, lf_p\} = LF$, $\{rf_1, \ldots, rf_q\} = RF$,
where $l'_{m'}, \ldots, l'_1$ is the sublist of $l_m, \ldots, l_1$ without elements in LF
where $r'_1, \ldots, r'_{n'}$ is the sublist of $r_1, \ldots, r_n$ without elements in RF.
$\lambda_H(w) := \lambda_H(w) \cup \{t_w\}$; // expansion
end end

Algorithm $\mathbf{TGE}^{(K)}_{disp}$ (type-generalize-expand) :
Input : $\sigma[i]$ (σ being a training sequence).
Output : GCD $\mathbf{TGE}^{(K)}_{disp}(\sigma[i])$.
let G_H be CDG with lexicon λ_H
$\lambda_H := \emptyset$;
(loop) **for** $i \geq 0$ //Infinite loop on σ
let $\sigma[i+1] = \sigma[i] \cdot D$;
let $(x, E) = D$; // D is the next example
(loop) **for every** $w \in x$ // w is a word in D
let $V(w, D) = [l_m \backslash \cdots \backslash l_1 \backslash h / r_1 / \cdots / r_n]^P$ // vicinity of w in D
let $LT := \{l_1\} \cup \cdots \cup \{l_m\}$
let $LF := \{d : d \in LT, card(\{i : 1 \leq i \leq m, l_i = d\}) \geq K\}$ // left K-repeatable
let $RT := \{r_1\} \cup \cdots \cup \{r_n\}$
let $RF := \{d : d \in RT, card(\{i : 1 \leq i \leq n, r_i = d\}) \geq K\}$ // right K-repeatable
let $t_w := [\{lf_1^*, \ldots, lf_p^*\} \backslash l'_{m'} \backslash \cdots \backslash l'_1 \backslash h / r'_1 / \cdots / r'_{n'} / \{rf_1^*, \ldots, rf_q^*\}]^P$ // generalised type
where $\{lf_1, \ldots, lf_p\} = LF$, $\{rf_1, \ldots, rf_q\} = RF$,
where $l'_{m'}, \ldots, l'_1$ is the sublist of $l_m, \ldots, l_1$ without elements in LF
where $r'_1, \ldots, r'_{n'}$ is the sublist of $r_1, \ldots, r_n$ without elements in RF.
$\lambda_H(w) := \lambda_H(w) \cup \{t_w\}$; // expansion
end end

TH $\mathbf{TGE}^{(K)}_{disp}$ incrementally learns dispersed K-star revealing CDG from DS.

Conclusions

- To express the repeatable dependencies in the traditional linguistics' perspective of dependencies as words' valencies one needs iteration
- Unification based inference methods fail when applied to categorial grammars extended by iteration :
  the rigid grammars are not learnable from structures
- K-star revealing CDG, complying with the RD Principle, are incrementally learnable from DS for three kinds of iteration : sequential, dispersed and choice, when used separately

Conclusions

- To express the repeatable dependencies in the traditional linguistics' perspective of dependencies as words' valencies one needs iteration
- Unification based inference methods fail when applied to categorial grammars extended by iteration :
  the rigid grammars are not learnable from structures
- K-star revealing CDG, complying with the RD Principle, are incrementally learnable from DS for three kinds of iteration : sequential, dispersed and choice, when used separately

Future Work

- To find a reasonable definition of K-star revealing for the CDG jointly using the three iterations

Conclusions

- To express the repeatable dependencies in the traditional linguistics' perspective of dependencies as words' valencies one needs iteration
- Unification based inference methods fail when applied to categorial grammars extended by iteration :
  the rigid grammars are not learnable from structures
- K-star revealing CDG, complying with the RD Principle, are incrementally learnable from DS for three kinds of iteration : sequential, dispersed and choice, when used separately

Future Work

- To find a reasonable definition of K-star revealing for the CDG jointly using the three iterations

THANK YOU !

CDG G_{target} is dispersed 2-star revealing and has lexicon :

$John \mapsto [N]$ $\quad$ $to\_the\_station \mapsto [L]$

$ran \mapsto [N\backslash A*\backslash S/A*/L/A*],\ [N\backslash A*\backslash S/A*]$

$seemingly,\ slowly,\ alone,\ during\_half\_an\_hour,\ every\_morning \mapsto [A]$

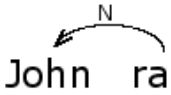
CDG G_{target} is dispersed 2-star revealing and has lexicon :

$John \mapsto [N]$ $to\_the\_station \mapsto [L]$

$ran \mapsto [N\backslash A*\backslash S/A*/L/A*],\ [N\backslash A*\backslash S/A*]$

$seemingly,\ slowly,\ alone,\ during\_half\_an\_hour,\ every\_morning \mapsto [A]$

Algorithm $\mathbf{TGE}^{(2)}_{disp}$ will add :

$ran \mapsto [N\backslash S]$ for 

.

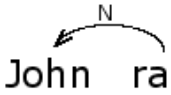
CDG G_{target} is dispersed 2-star revealing and has lexicon :

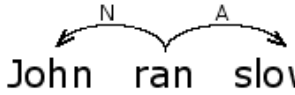
$John \mapsto [N]$ $to\_the\_station \mapsto [L]$

$ran \mapsto [N\backslash A*\backslash S/A*/L/A*],\ [N\backslash A*\backslash S/A*]$

$seemingly,\ slowly,\ alone,\ during\_half\_an\_hour,\ every\_morning \mapsto [A]$

Algorithm $\mathbf{TGE}^{(2)}_{disp}$ will add :

$ran \mapsto [N\backslash S]$ for 

. $ran \mapsto [N\backslash S/A]$ for 

.

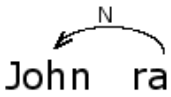
CDG G_{target} is dispersed 2-star revealing and has lexicon :

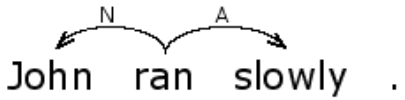
John $\mapsto [N]$ *to_the_station* $\mapsto [L]$

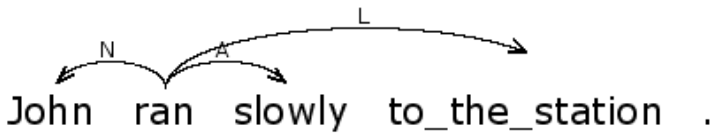
ran $\mapsto [N\backslash A*\backslash S/A*/L/A*],\ [N\backslash A*\backslash S/A*]$

seemingly, *slowly*, *alone*, *during_half_an_hour*, *every_morning* $\mapsto [A]$

Algorithm $\mathbf{TGE}^{(2)}_{disp}$ will add :

ran $\mapsto [N\backslash S]$ for 

ran $\mapsto [N\backslash S/A]$ for 

ran $\mapsto [N\backslash S/L/A]$ for 

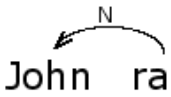
CDG G_{target} is dispersed 2-star revealing and has lexicon :

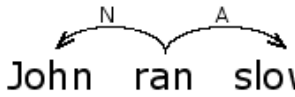
John $\mapsto [N]$ *to_the_station* $\mapsto [L]$

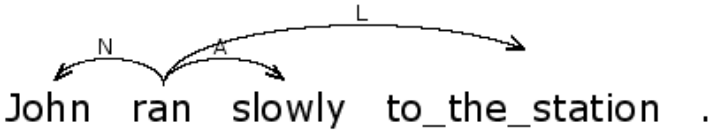
ran $\mapsto [N\backslash A*\backslash S/A*/L/A*],\ [N\backslash A*\backslash S/A*]$

seemingly, *slowly*, *alone*, *during_half_an_hour*, *every_morning* $\mapsto [A]$

Algorithm $\mathbf{TGE}^{(2)}_{disp}$ will add :

ran $\mapsto [N\backslash S]$ for 

ran $\mapsto [N\backslash S/A]$ for 

ran $\mapsto [N\backslash S/L/A]$ for 

ran $\mapsto [N\backslash A\backslash S/\{A*\}]$ for 

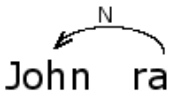
CDG G_{target} is dispersed 2-star revealing and has lexicon :

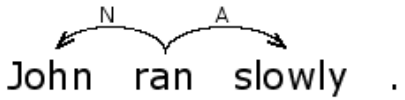
$John \mapsto [N]$ $to\_the\_station \mapsto [L]$

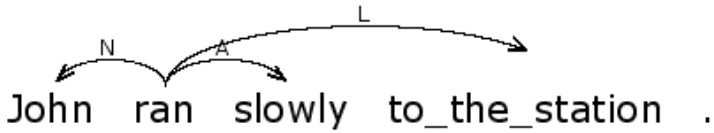
$ran \mapsto [N\backslash A* \backslash S/A* /L/A*],\ [N\backslash A* \backslash S/A*]$

$seemingly,\ slowly,\ alone,\ during\_half\_an\_hour,\ every\_morning \mapsto [A]$

Algorithm $\mathbf{TGE}^{(2)}_{disp}$ will add :

$ran \mapsto [N\backslash S]$ for 

$ran \mapsto [N\backslash S/A]$ for 

$ran \mapsto [N\backslash S/L/A]$ for 

$ran \mapsto [N\backslash A\backslash S/\{A*\}]$ for 

etc... $ran \mapsto [N\backslash A\backslash S/L/\{A*\}]$ for