# Two models of learning iterated dependencies

Denis Béchet, Alexander Dikovsky and Annie Foret

LINA CNRS UMR 6241, Université de Nantes and IRISA, Université de Rennes1, France

# Plan

1. Categorial dependency grammars (CDG)
2. Learning from positive data
3. Unlearnability of rigid CDG from FA-structures
4. Incremental learning of CDG from DS
5. Conclusion

# 1. Categorial dependency grammars

- are type logical grammars (categorial grammars with FO types),
- define unlimited DS (including discontinuous and long distance dependencies),
- are lexicalized and completely local (grammars of word's valencies),
- are adapted to flexible order (extended by flat regular expressions),
- are polynomially parsed (efficient deterministic offline parser is implemented),
- are well suited for practical use (cf. a wide scope large scale CDG of French).

# Types of projective dependencies

$Gov \xrightarrow{d} Sub$ is typed as :

$Gov \mapsto [..\backslash../../d/..]$ for the governor

and as

$Sub \mapsto [..\backslash d/..]$ for the subordinate.

**EX:**


$in \mapsto [c\_copul/prepos-in]$ $the \mapsto [det]$ $beginning \mapsto [det\backslash prepos-in]$

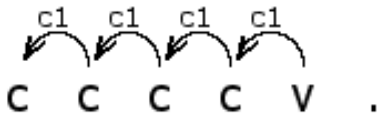
$was \mapsto [c\_copul\backslash S/pred]$ $Word \mapsto [det\backslash pred]$

# Necessity of iterated types

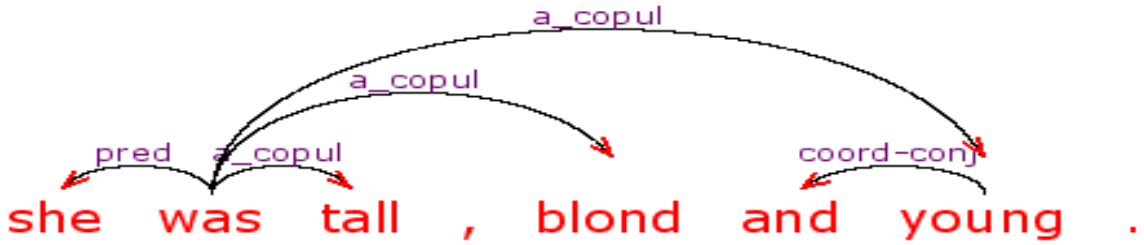
Specific interpretation of subtype elimination:

$a \mapsto [\alpha\backslash d]$, $b \mapsto [d\backslash\beta]$ derives the dependency $a \overset{d}{\longleftarrow} b$ for $ab$.
Therefore, the recursive types derive sequenced dependencies:

$v \mapsto [c1\backslash S]$, $c \mapsto [c1\backslash c1], [c1]$ derives for $ccccv$ the DS:

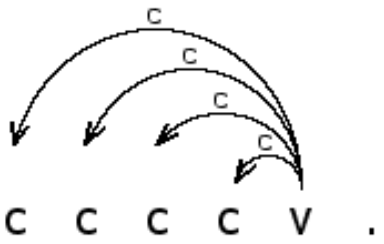



Principle of optional repeatable dependencies (see [Mel'cuk'88]):
- modifiers of a noun $n$ share the governor $n$,
- circumstantials of a verb $v$ share the governor $v$.




This is why the explicit iterated types are needed:

$v \mapsto [c*\backslash S]$, $c \mapsto [c]$ derives for $ccccv$ the DS:



# Types of non-projective dependencies

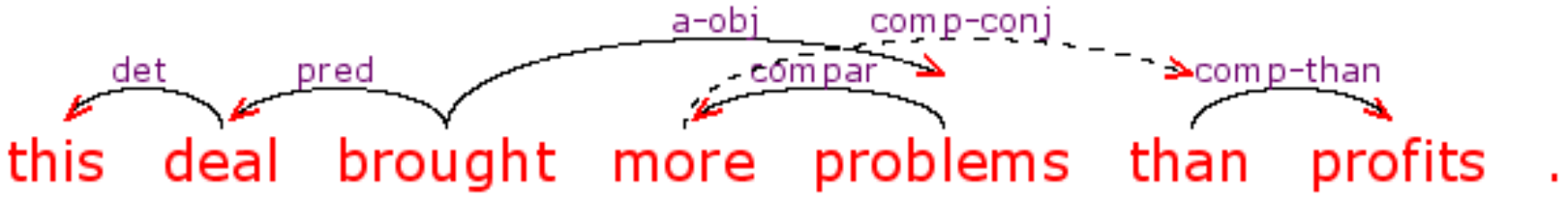
are defined using dual polarized valencies:

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

$Gov \mapsto [..]^{\nearrow d}$ for the governor

and

$Sub \mapsto [..]^{\searrow d}$ for the subordinate.




$this \mapsto [det]$ $deal \mapsto [det\backslash pred]$ $brought \mapsto [pred\backslash S/a-obj]$

$problems \mapsto [compar\backslash a-obj]$ $profits \mapsto [comp-than]$

$more \mapsto [compar]^{\nearrow comp\text{–}conj}$ $than \mapsto [/comp-than]^{\searrow comp\text{–}conj}$

# CDG calculus

$\mathbf{L^l}$. $C^{P_1}[C\backslash\beta]^{P_2} \vdash [\beta]^{P_1P_2}$

$\mathbf{I^l}$. $C^{P_1}[C^*\backslash\beta]^{P_2} \vdash [C^*\backslash\beta]^{P_1P_2}$

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

$\mathbf{D^l}$. $\alpha^{P_1}(\swarrow C)P(\nwarrow C)P_2 \vdash \alpha^{P_1PP_2}$,

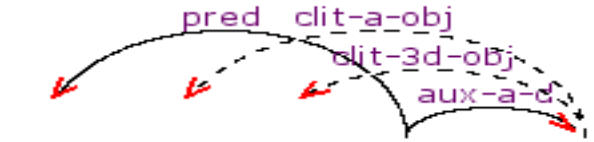
if $(\swarrow C)P(\nwarrow C)$ satisfies the following valency pairing principle:

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 $\swarrow C, \nwarrow C$.
Similar for right valencies.

Elimination of dual valencies $v = (\swarrow C)$, $\breve{v} = (\nwarrow C)$ by the rule $\mathbf{D^l}$ creates the *discontinuous dependency* $C$.

# Proofs in CDG calculus

EX: French clitics




(Fr. $^{*}She\ it_{g=f}\ him\ has\ given$)

*elle* $\mapsto [pred]$

*la* $\mapsto [\varepsilon]^{\swarrow clit\text{–}a\text{–}obj}$

*lui* $\mapsto [\varepsilon]^{\swarrow clit\text{–}3d\text{–}obj}$

*a* $\mapsto [pred \backslash S / aux]$

*donnée* $\mapsto [aux]^{\nwarrow clit\text{–}3d\text{–}obj \nwarrow clit\text{–}a\text{–}obj}$

$$[pred]\ [\varepsilon]^{\swarrow clit\text{–}dobj}\ [\varepsilon]^{\swarrow clit\text{–}iobj}\ [pred \backslash S / aux]\ [aux]^{\nwarrow clit\text{–}iobj, \nwarrow clit\text{–}dobj} \vdash S$$

$$\dfrac{[pred]\quad \dfrac{[\varepsilon]^{\swarrow clit\text{–}dobj}\quad \dfrac{[\varepsilon]^{\swarrow clit\text{–}iobj}\ [pred \backslash S / aux]}{[pred \backslash S / aux]^{\swarrow clit\text{–}iobj}}\ (\mathbf{L}^l)}{[pred \backslash S / aux]^{\swarrow clit\text{–}dobj \swarrow clit\text{–}iobj}}\ (\mathbf{L}^l)}{[S / aux]^{\swarrow clit\text{–}dobj \swarrow clit\text{–}iobj}}\ (\mathbf{L}^{\mathbf{l}})$$

$$\dfrac{[S]^{\swarrow clit\text{–}dobj \swarrow clit\text{–}iobj \nwarrow clit\text{–}iobj \nwarrow clit\text{–}dobj}}{S}\ (\mathbf{D}^l \times 2)$$

$$\dfrac{[aux]^{\nwarrow clit\text{–}iobj \nwarrow clit\text{–}dobj}}{[S]^{\swarrow clit\text{–}dobj \swarrow clit\text{–}iobj \nwarrow clit\text{–}iobj \nwarrow clit\text{–}dobj}}\ (\mathbf{L}^r)$$

# 2. Learning from positive data

# Learning in the limit

Grammatical inference in a family $\mathcal{C}$ [E. Gold'67]:

With every grammar $G \in \mathcal{C}$ is related an observation set $\Phi(G)$ of $G$ ($\Phi(G)$ is $L(G)$ or a structure language generated by $G$).
$A$ is an inference algorithm for $\mathcal{C}$ if:

- for every grammar $G \in \mathcal{C}$, $A$ applies to the training sequences for $G$, i.e. to enumerations $\sigma$ of $\Phi(G)$
- for every initial subsequence $\sigma[i] = \{s_1, \ldots, s_i\}$ of $\sigma$, it returns a hypothetical grammar $A(\sigma[i]) = G_i \in \mathcal{C}$.

$A$ learns a target grammar $G \in \mathcal{C}$ if on any training sequence $\sigma$ for $G$ $A$ stabilizes on a grammar $\mathcal{A}(\sigma[T]) \equiv G$. [a]

The grammar $\lim_{i \to \infty} \mathcal{A}(\sigma[i]) = \mathcal{A}(\sigma[T])$ returned at the stabilization step $T$ is the limit grammar.
$A$ learns $\mathcal{C}$ if it learns every grammar in $\mathcal{C}$. $\mathcal{C}$ is learnable if there is an inference algorithm learning $\mathcal{C}$.

---

[a] $\mathcal{A}$ stabilizes on $\sigma$ on step $T$: $T$ is the minimal number $t$ for which there is no $t_1 > t$ such that $\mathcal{A}(\sigma[t_1]) \neq \mathcal{A}(\sigma[t])$.

# Unlearnability

Many well known grammatical families are unlearnable from strings because they generate limit points:

**LIMIT POINTS**: A class $\mathcal{L}$ of languages has a limit point iff there is an infinite sequence of languages $(L_n)_{n \in N} \in \mathcal{L}$ and a language $L \in \mathcal{L}$ such that: $L_0 \subsetneq L_1 \ldots \subsetneq \ldots \subsetneq L_n \subsetneq \ldots$ and $L = \bigcup_{n \in N} L_n$.

[K.Wright,T.Motoki,T.Shinohara'89,91]: If $\mathcal{L}(\mathcal{G})$ has a limit point, then $\mathcal{G}$ is unlearnable from strings.

**COR**: A family generating all finite languages and at least one infinite language is unlearnable from strings (CDG are also unlearnable for this reason).

**INFINITE ELASTISITY**: A class $\mathcal{L}$ of languages has infinite elasticity if there are infinite sequences of sentences $(e_i)_{i \in \mathbb{N}}$ and of languages $(L_i)_{i \in \mathbb{N}}$ in $\mathcal{L}$ such that $e_i \notin L_i$ and $\{e_0, \ldots, e_{i-1}\} \subseteq L_i$ for all $i$.

[K.Wright'89]: Every unlearnable family $\mathcal{L}$ has infinite elasticity.

# Learnability

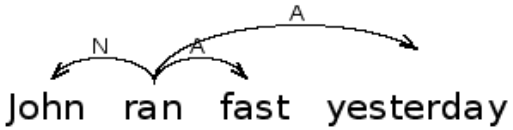
1. From strings: Finite thickness $\Rightarrow$ Finite elasticity [T.Shinohara'91], Finite elasticity $\Rightarrow$ learnability from strings [K.Wright'89].

**EX**: $k$-rule string and term generating grammars.

2. From structures: For categorial grammars, FA-structures and the like are terms partially defining the proofs.

**EX**: For CDG $\lambda(John)=N$, $\lambda(ran)=[N\setminus S/A^*]$, $\lambda(fast)=\lambda(yesterday)=A$,

$$\mathcal{D}:\quad \dfrac{\dfrac{\dfrac{[N\setminus S/A^*]\ A}{[N\setminus S/A^*]}\mathbf{L^r}\qquad A}{[N\setminus S/A^*]}\mathbf{L^r}}{\dfrac{N\qquad [N\setminus S]}{S}\mathbf{\Omega^r}}\mathbf{L^l}$$




the proof $\mathcal{D}$ is represented by the FA-structures:

$\mathbf{L^l}(John, \mathbf{L^r}(\mathbf{L^r}(ran, fast), yesterday))$ (unlabeled)

$\mathbf{L^l_N}(John, \mathbf{L^r_A}(\mathbf{L^r_A}(ran, fast), yesterday))$ (labeled)

Rigid CG (one type per word) are learnable from unlabeled structures by an unification based algorithm [W.Buszkowski & G.Penn'90, M.Kanazawa'98]

$\Rightarrow$ Rigid (and $k$-rigid) CG are learnable from strings [M.Kanazawa'98]

# 3. Unlearnability of rigid CDG from FA-structures

# Limit points caused by iteration

Even rigid CDG are not learnable from strings: limit points can be defined using iterated types. Rigid CDG without iteration are learnable from unlabeled FA-structures[D.Béchet & A.Dikovsky & A.Foret & E.Moreau'04]

The following rigid CDG define a limit point:

$$C'_0 = S \qquad C'_{n+1} = C'_n \,/\, A^* \,/\, B^*$$

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

Let
$flat_{\mathbf{L^r}}(w) =$ for $w \in \{a, b, c\}$,
$flat_{\mathbf{L^r}}(x.w) = \mathbf{L^r}(flat_{\mathbf{L^r}}(x), w)$ for $x \in \{a, b, c\}^*$ and $w \in \{a, b, c\}$.

Let $FL(G)$ denote the language of FA-structures of $G$.

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

**COR** Rigid CDG are not learnable from unlabeled FA-structures.

One cannot do without iterated types and without multiple type assignments in real application CDGs.

So we will try another model . . .

# 4. Incremental learning of CDG from DS

# Flexibility order

**NOTATION:** For a type $t = [l_m \backslash \cdots l_1 \backslash g / r_1 \cdots / r_n]^P$ and a dependency $c$,

$$t_c^{(i\backslash,j)} =_{df} [l_m \backslash \cdots \backslash l_j \backslash c \cdots \backslash c \backslash l_{j-1} \backslash \cdots l_1 \backslash g / r_1 \cdots / r_n]^P \ (i \text{ times})$$

(similar for right arguments).

$$t_c^{(*\backslash,j)} =_{df} [l_m \backslash \cdots \backslash l_j \backslash c* \backslash l_{j-1} \backslash \cdots l_1 \backslash g / r_1 \cdots / r_n]^P$$

Flexibility order $G_1 \preceq G_2$ intuitively means that $G_2$ defines no less DS than $G_1$ and at least as precise DS as $G_1$. $\preceq$ is the reflexive-transitive closure of the following preorder $<$:

1. $t_c^{(i\backslash,j)} < t_c^{(*\backslash,j)}$ and $t_c^{(i/,k)} < t_c^{(*/,k)}$ for all $i \geq 0$, $0 \leq j \leq m$ and $0 \leq k \leq n$
2. $\tau < \tau'$ for sets of types $\tau, \tau'$, if either:

$(i)$ $\tau' = \tau \cup \{t\}$ for a type $t \notin \tau$ or

$(ii)$ $\tau = \tau_0 \cup \{t'\}$ and $\tau' = \tau_0 \cup \{t''\}$

for a set of types $\tau_0$ and some types $t', t''$ such that $t' < t''$.

3. $\lambda < \lambda'$ for two type assignments $\lambda$ and $\lambda'$, if $\lambda(w') < \lambda'(w')$ for a word $w'$ and $\lambda(w) = \lambda'(w)$ for all words $w \neq w'$.

# Incremental learning

**PROPOSITION** If $G_1 \preceq G_2$, then $\Delta(G_1) \subseteq \Delta(G_2)$ and $L(G_1) \subseteq L(G_2)$.

Incremental learning: Let $\mathcal{A}$ be an inference algorithm for $\mathcal{CDG}$ from DS and $\sigma$ be a training sequence for a CDG $G$. Then:

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

**THEOREM** Let $\sigma$ be a training sequence for a CDG $G$. If an inference algorithm $\mathcal{A}$ is monotonic, faithful, and expansive on $\sigma$, and if $\mathcal{A}$ stabilizes on $\sigma$ then $\lim_{i \to \infty} \mathcal{A}(\sigma[i]) \equiv_s G$.

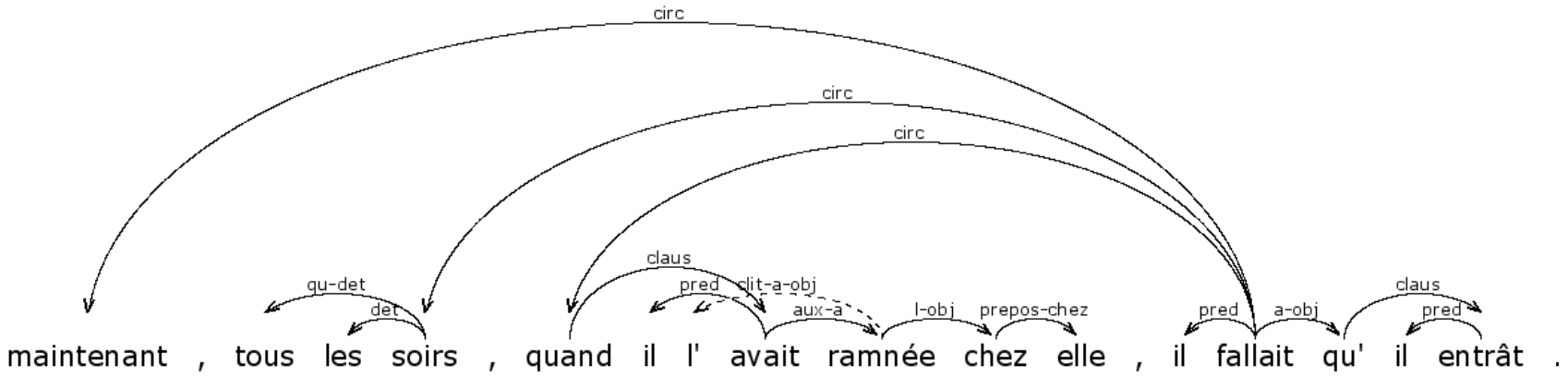
# K-star revealing CDG. Main notions-1

1. Repetition blocks (R-blocks): for a dependency $d \in \mathbf{C}$,

$$LB_d = \{x_1 \backslash \cdots \backslash x_i \parallel i > 0, \; x_i \in \{d, d^*\}\}, RB_d = \{x_1 / \cdots / x_i \parallel i > 0, \; x_i \in \{d, d^*\}\}$$

2. Patterns: defined exactly as types, but in the place of elementary dependencies $\mathbf{C}$ are used gaps: $\mathbf{G} = \{<d> \parallel d \in \mathbf{C}\}$.
Constraint: $[\alpha \backslash <d> \backslash <d> \backslash \beta]^P$, $[\alpha / <d> / <d> / \beta]^P$ are not patterns.

3. Vicinity of a word $w$ in a DS $D$ is the $\preceq$-minimal type $t^P$ defining all dependencies of $w$ in $D$.




(fr. *now all the evenings when he took her home he had to enter [M.Proust])

The verb *fallait* in this DS has the vicinity $[pred \backslash circ \backslash circ \backslash circ \backslash S / a-obj]$

# K-star revealing CDG. Main notions-2

Superposition: $(i)$ For a pattern $\pi$ and R-blocks $\beta_1, \ldots, \beta_k$,

$$\pi(<d_1>\leftarrow \beta_1, \ldots, <d_k>\leftarrow \beta_k)$$

is the result of parallel substitution into $\pi$ of $\beta_i$ for the corresponding $<d_i>$.
$(ii)$ $\pi$ is superposable on a type or a vicinity $E$ if:

$$E = \pi(<d_1>\leftarrow \beta_1, \ldots, <d_k>\leftarrow \beta_k)$$

where $\beta_1, \ldots, \beta_k$ are R-blocks and every $\beta_i$ is maximal length in $E$.

**PROPOSITION** For every type (vicinity) $E$ there is a single pattern $\pi$ superposable on $E$ and a single decomposition (R-decomposition)

$$E = \pi(<d_1>\leftarrow \beta_1, \ldots, <d_k>\leftarrow \beta_k)^P.$$

R-block substitution: Let $E$ be a type or a vicinity with the R-decomposition: $E = \pi(<d_1>\leftarrow \beta_1, \ldots, <d_k>\leftarrow \beta_k)^P$. Then:

$$E[i \leftarrow \beta] =_{df} \pi(<d_1>\leftarrow \beta_1, \ldots, <d_i>\leftarrow \beta, \ldots, <d_k>\leftarrow \beta_k)^P.$$

**EX:** The pattern superposable on the vicinity $E = [pred\backslash circ\backslash circ\backslash circ\backslash S/a\text{–}obj]$ of *fallait* is $\pi = [<pred> \backslash <circ> \backslash S/ <a-obj>]$. Each of the assignments:

*fallait* $\mapsto \pi[1 \leftarrow pred, 2 \leftarrow circ\backslash circ\backslash circ, 3 \leftarrow a-obj]$,

*fallait* $\mapsto \pi[1 \leftarrow pred, 2 \leftarrow circ*, 3 \leftarrow a-obj]$ and

*fallait* $\mapsto \pi[1 \leftarrow pred, 2 \leftarrow circ*circ, 3 \leftarrow a-obj]$ may define $E$.

# K-star revealing CDG. Definition

Let $G$ be a CDG with lexicon $\lambda$ and $t$ be a type (vicinity) with R-decomposition $t = \pi(< d_1 > \leftarrow \beta_1, \ldots, < d_k > \leftarrow \beta_k)^P$. Then $G_w^{t[i \leftarrow \beta]}$ denotes the CDG with lexicon $\lambda \cup \{w \mapsto t[i \leftarrow \beta]\}$.

**DEFINITION:** Let $K > 1$ be an integer. CDG $G$ is $K$-star revealing for dependency $d$ if

$$G_w^{t[i \leftarrow d^*]} \equiv_s G$$

for every word $w$ and every $t$ which is a type $t \in \lambda(w)$ or a vicinity of $w$ in $\Delta(G)$, such that:

- $t = t[i \leftarrow \beta]$ and
- $\beta$ has at least one occurrence of $d^*$ or at least $K$ occurrences of $d$.

$G$ is $K$-star revealing if it is $K$-star revealing for all dependencies.

**EX:** Let $G(t)$ be $A \mapsto [a], B \mapsto [b], C \mapsto t$ for a type $t$. Then:

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

# Incremental learnability

**THEOREM** The class $\mathcal{CDG}^{K\rightarrow *}$ of $K$-star revealing CDG is incrementally learnable from DS.

**Algorithm** $\mathbf{TGE}^{(K)}$ (type-generalize-expand):
**Input**: $\sigma[i]$ ($\sigma$ being a training sequence).
**Output**: CDG $\mathbf{TGE}^{(K)}(\sigma[i])$.
**let** $G_H = (W_H, \mathbf{C}_H, S, \lambda_H)$ where $W_H := \emptyset$; $\mathbf{C}_H := \{S\}$; $\lambda_H := \emptyset$;
(loop) **for** $i \geq 0$ //Infinite loop on $\sigma$
**let** $\sigma[i+1] = \sigma[i] \cdot D$;
**let** $(x, E) = D$;
(loop) **for every** $w \in x$
$W_H := W_H \cup \{w\}$;
**let** $V(w, D) = \pi(<d_1>\leftarrow \beta_1, \ldots, <d_k>\leftarrow \beta_k)^P$ (vicinity of $w$ in $D$)
(loop) **for** $j := 1, \ldots, k$
**if** $\beta_j \in LD_d \cup RD_d$ **and** $length(\beta_j) \geq K$
**then** $\gamma_j := d^*$ // generalization
**else** $\gamma_j := \beta_j$ **end end**
**let** $t_w := \pi(<d_1>\leftarrow \gamma_1, \ldots, <d_k>\leftarrow \gamma_k)^P$ // typing
$\lambda_H(w) := \lambda_H(w) \cup \{t_w\}$; // expansion
**end end**

**LEMMA** $\mathbf{TGE}^{(K)}$ is monotonic, faithful and expansive on every training sequence $\sigma$ of a $K$-star revealing CGD.

**LEMMA** $\mathbf{TGE}^{(K)}$ stabilizes on every training sequence $\sigma$ of a $K$-star revealing CGD.

# Example

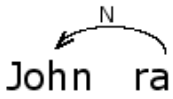
Lexicon of 2-star-revealing CDG $G_{target}$ contains assignments:

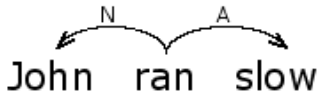
$John \mapsto [N]$ $\quad to\_the\_station \mapsto [L]$

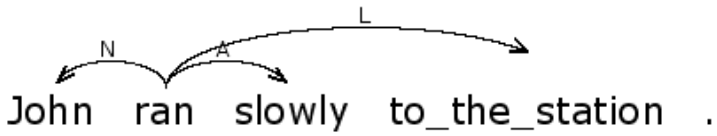
$ran \mapsto [N\backslash A*\backslash S/A*/L/A*],\ [N\backslash A*\backslash S/A*]$

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

Algorithm $\mathbf{TGE}^{(2)}(\sigma[i])$ will add:

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

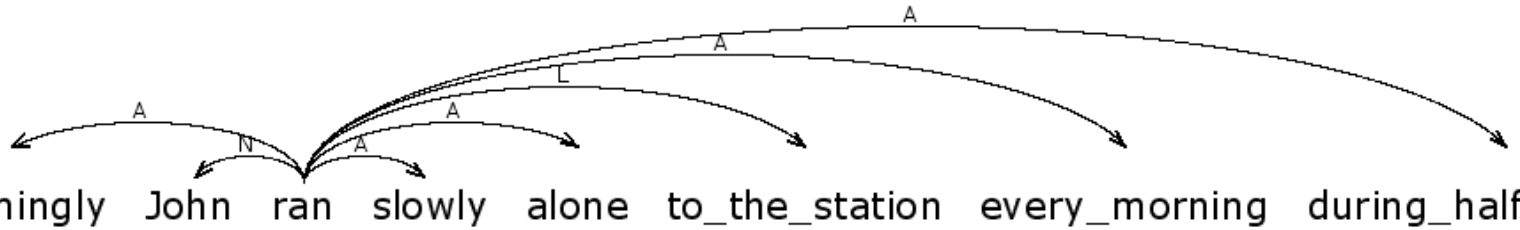

$ran \mapsto [N\backslash S/A]$ for 


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


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


etc...

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



# 5. Conclusion

- Linguistically appropriate CDG are not rigid and need iterated types
- Traditional unification based learning from FA-structures fails even for rigid categorial dependency grammars with iterated types
- We propose a new model of incremental learning of $K$-star revealing CDG with iterated types from input DS without marked iteration
- $K$-star-revealing property is widely accepted in traditional linguistics for small $K$, which makes this model interesting for practical purposes
- The new model reflects the real situation of deterministic inference of a dependency grammar from a dependency treebank