Categorial Dependency Grammars: Analysis and Learning (Invited Talk)

Denis Béchet, University of Nantes
Annie Foret, University Rennes 1, France

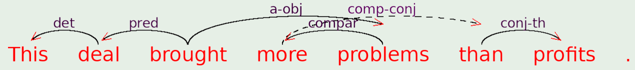
LACompLing 2021, Montpellier, December 15–17 2021

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

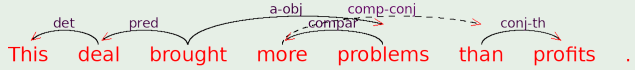
Saturation of valency pred determines projective dependency

deal $\xleftarrow{\text{pred}}$ *brought* (**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

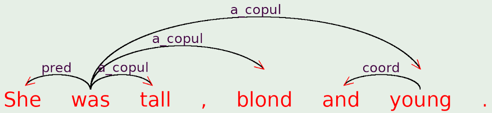
more has +valency comp-conj of a word somewhere on its right

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

Saturation of comp-conj determines non-projective dependency

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

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

CDG Types express dependency valencies

PROJECTIVE DEPENDENCIES (AND ANCHORS)

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

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

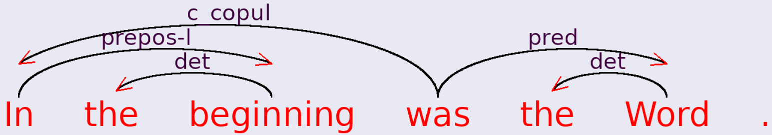
Subordinate Type: $Sub \mapsto [..\backslash d/..]^P$

Anchors are non-important projective dependencies. Used for:

- Anchoring punctuation
- Anchoring the subordinate of non-projective dependencies

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies



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

CDG Types express dependency valencies

NON-PROJECTIVE DEPENDENCIES

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

Dependency: $Gov \overset{d}{\dashrightarrow} Sub$:

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

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

CDG Types express dependency valencies

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

CDG Types express dependency valencies

NON-PROJECTIVE DEPENDENCIES WITH ANCHORS

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

Anchor valencies: $\#\searrow d$, $\#\swarrow d$

Dependency and anchor: $Gov \overset{d}{\dashrightarrow} Sub \overset{\#\searrow d}{\longleftarrow} Host$:

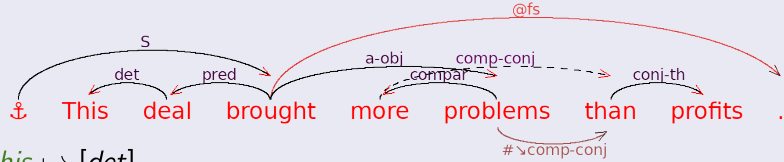
Governor Type: $Gov \mapsto [..]^{..\nearrow d..}$

Subordinate Type: $Sub \mapsto [..\backslash\#\searrow d/..]^{..\searrow d..}$

Host Type: $Host \mapsto [..\backslash\#\searrow d\backslash../..]^{P}$

Categorial Dependency Grammars (CDG)

CDG Types express dependency valencies



$this \mapsto [det]$

$deal \mapsto [det\backslash pred]$

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

$problems \mapsto [compar\backslash a-obj/\#\searrow comp-conj]$

$profits \mapsto [conj-th]$

$more \mapsto [compar]^{\nearrow comp-conj}$

$than \mapsto [\#\searrow comp-conj/conj-th]^{\searrow comp-conj}$

$. \mapsto [@fs]$

Left-oriented rules

L^{l}. $[C]^P[C\backslash\beta]^Q \vdash [\beta]^{PQ}$ $\quad Gov \xrightarrow{C} Sub$

Left-oriented rules

L^{l}. $[C]^{P}[C\backslash\beta]^{Q} \vdash [\beta]^{PQ}$ $\qquad Gov \xrightarrow{C} Sub$

$\mathrm{L}^{\mathrm{l}}_{\varepsilon}$. $[\ ]^{P}[\beta]^{Q} \vdash [\beta]^{PQ}$ $\qquad$ (no new dependency)

Left-oriented rules

L^{l}. $[C]^P[C\backslash\beta]^Q \vdash [\beta]^{PQ}$ $\qquad Gov \xrightarrow{C} Sub$

$\mathrm{L}^{\mathrm{l}}_{\varepsilon}$. $[\ ]^P[\beta]^Q \vdash [\beta]^{PQ}$ $\qquad$ (no new dependency)

I^{l}. $[C]^P[C^*\backslash\beta]^Q \vdash [C^*\backslash\beta]^{PQ}$ $\qquad Gov \xrightarrow{C} Sub$

Ω^{l}. $[C^*\backslash\beta]^P \vdash [\beta]^P$ $\qquad$ (no new dependency)

Left-oriented rules

L^{l}. $[C]^P[C\backslash\beta]^Q \vdash [\beta]^{PQ}$ $\qquad Gov \xrightarrow{C} Sub$

$\mathrm{L}^{\mathrm{l}}_{\varepsilon}$. $[\ ]^P[\beta]^Q \vdash [\beta]^{PQ}$ (no new dependency)

I^{l}. $[C]^P[C^*\backslash\beta]^Q \vdash [C^*\backslash\beta]^{PQ}$ $\qquad Gov \xrightarrow{C} Sub$

Ω^{l}. $[C^*\backslash\beta]^P \vdash [\beta]^P$ (no new dependency)

D^{l}. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1PP_2}$ $\qquad Gov \overset{C}{\dashrightarrow} Sub$

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 $\swarrow C, \nwarrow C$

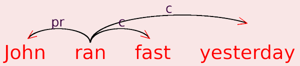
LEXICON:

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

Derivation

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

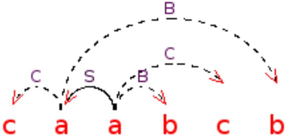
Dependency structures



L^l	$[C]^P[C\backslash\beta]^Q \vdash [\beta]^{PQ}$	L^r	$[\beta/C]^P[C]^Q \vdash [\beta]^{PQ}$
I^l	$[C]^P[C*\backslash\beta]^Q \vdash [C*\backslash\beta]^{PQ}$	I^r	$[\beta/C*]^P[C]^Q \vdash [\beta/C*]^{PQ}$
Ω^l	$[C*\backslash\beta]^P \vdash [\beta]^P$	Ω^r	$[\beta/C*]^P \vdash [\beta]^P$
D^l	$\alpha^{P_1(\swarrow V)P(\nwarrow V)P_2} \vdash \alpha^{P_1 P P_2}$, if FA	D^r	$\alpha^{P_1(\nearrow V)P(\searrow V)P_2} \vdash \alpha^{P_1 P P_2}$, if FA

CDG example: mix [LACL2005]

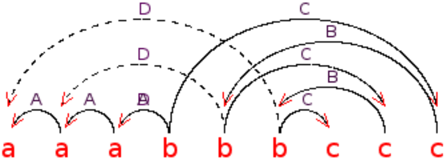
```
a  [S]<↖B,↖C>
   [S\S]<↖B,↖C>
   [S]<↗C,↗B>T
   [S\S]<↗C,↗B>
   [S]<↖B,↗C>
   [S\S]<↖B,↗C>
   [S]<↖C,↗B>
   [S\S]<↖C,↗B>
b  [ ]<↙B>
   [ ]<↘B>
c  [ ]<↙C>
   [ ]<↘C>
```



A CDG for mix with a parse example

In the above grammar, some types have empty heads ; other grammars avoiding empty heads can be provided, but the above one is simpler.

a	[A]<↙D>
	[A\A]<↙D>
b	[B/C]<↖D>
	[A\S/C]<↖D>
c	[C]
	[B\C]



A CDG for $a^n b^n c^n$ with a parse example

Parsing Algorithms

CdgAnalyst (Dekhtyar-Dikovsky-Karlov, TCS 2015)

- Dynamic programming parsing algorithm
- Based on CYK parsing algorithm

  $+$ polarized valency calculus information

Filtering parsers (Alfared-Béchet-Dikovsky, Depling 2011)

- Reduction of the search space
- Based on sentence "complexity" of natural languages

  $\Longrightarrow$ Limit the complexity of potentials

Greedy parsers (Lacroix-Béchet, Coling 2014)

- Transition-Based Dependency Parser
- 3 steps (local / left non-projective / right non-projective)

Theorem 8

Algorithm **CdgAnalyst** *has time complexity*

$$\mathbf{O}\left(l_G \cdot a_G^2 \cdot (\Delta_G \cdot n)^{2p_G} \cdot n^3\right).$$

Complexity of CdgAnalyst (Dekhtyar-Dikovsky-Karlov, TCS 2015)

n : The length of the input string
l_G : The number of assignments in G
a_G : The maximal number of left or right subtypes in G
Δ_G : The maximal valency deficit in G
p_G : The number of polarized valency names in G

Plan

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**: $\sigma(\mathbb{N}) = L(G_T)$

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

from strings (FG'2004)

- k-valued CDG **without** $*$ iterated types are learnable from structures and from strings
- rigid CDG **with** $*$ are not learnable from strings (a limit point).

Limit point

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

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

from structures (FG'2010, ...)

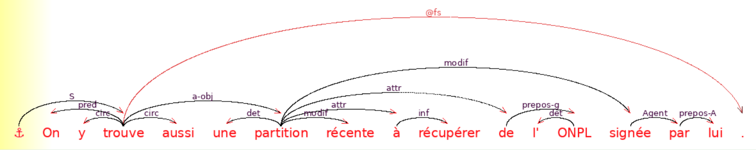
- rigid CDG **with** $*$ are not learnable from DS

So the CDG are not learnable from dependency treebanks !

Type-Generalize-Expand (TGE)

Computes words' types from their VICINITIES in DS

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



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

Type-Generalize-Expand (TGE) : types with d^*, repeating principle

Type-Generalize-Expand (TGE) : lexicon level, CV for a subclass

Algorithm TGE$^{(K)}$ (type-generalize-expand):
Input: σ, a training sequence of length N.
Output: CDG TGE$^{(K)}(\sigma)$.

let $G_H = (W_H, \mathrm{C}_H, S, \lambda_H)$ where $W_H := \emptyset$; $\mathrm{C}_H := \{S\}$; $\lambda_H := \emptyset$;
(loop) **for** $i = 1$ **to** N // loop on σ
  let D such that $\sigma[i] = \sigma[i-1] \cdot D$; // the i-th DS of σ
  let $(X, E) = D$;
  (loop) **for every** $w \in X$ // the order of the loop is not important
    $W_H := W_H \cup \{w\}$;
    let $t_w = V(w, D)$ // the vicinity of w in D
    (loop) **while** $t_w = [\alpha\backslash l\backslash \mathrm{d}\backslash \cdots \backslash \mathrm{d}\backslash r\backslash\beta]$
      with at least K consecutive occurrences of d, $l \neq d$ (or not present) and $r \neq d$ (or not present)
        $t_w := [\alpha\backslash l\backslash \mathrm{d}^*\backslash r\backslash\beta]$
    (loop) **while** $t_w = [\alpha/l/\mathrm{d}/\cdots/\mathrm{d}/r/\beta]$
      with at least K consecutive occurrences of d, $l \neq d$ (or not present) and $r \neq d$ (or not present)
        $t_w := [\alpha/l/\mathrm{d}^*/r/\beta]$
    $\lambda_H(w) := \lambda_H(w) \cup \{t_w\}$; // lexicon expansion
    end end
return G_H

TH TGE$^{(K)}$ learns *K-star revealing* CDG from DS (FG 2010)

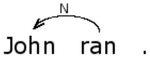
Importantly, no bound on the number of types is assumed

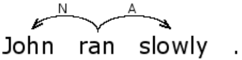
Exemple

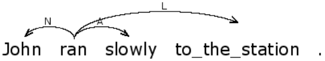
$John \mapsto [N]$ $to\_the\_station \mapsto [L]$
$ran \mapsto [N\backslash A^*\backslash S/A^*/L/A^*],\ [N\backslash A^*\backslash S/A^*]$
$seemingly,\ slowly,\ alone,\ every\_morning \mapsto [A]$

(Global Simple K-star)

Algorithm $\mathrm{TGE}^{(2)}$:

$ran \mapsto [N\backslash S]$ for $(i = 1)$: 

$ran \mapsto [N\backslash S/A]$ for $(i = 2)$: 

$ran \mapsto [N\backslash S/L/A]$ for $(i = 3)$: 

$ran \mapsto [N\backslash A\backslash S/\boxed{A^*}]$ for $(i = 4)$: 

...

Learning approaches with iterated types

Structured Example	Annotation	Number (k) of Types per word	Repetition number (K) for Indiscernibility
functor-argument (FA, proof-tree)	unlabelled (no dep. name)	bound	no bound
dependency structure (DS)	labelled (dep. names)	no bound	bound

from [Béchet-Foret, Machine Learning, 2021]

Criteria and readings of the "repetition principle"

consecutive or dispersed in a type ; left-right ; global

- K-star revealing (complex equivalence property)
- $\supseteq$ Simple K-star (syntactic) : (1) at most $K-1$ occurrences of d and (2) no occurrence of d if there exists at least one occurrence of d^* in each $l_1 \backslash l_2 \backslash ... l_p \backslash$ where each l_i is either d or some x^*
- $\supseteq$ Global Simple K-star : (1) (2) in each $l_1 \backslash l_2 \backslash ... l_p \backslash$ (both sides)

UD Corpora
also available for under-resourced languages (breton)

- producing a CDG grammar
- properties of some dependencies, repeating patterns, measures

Star Scope	Star Pattern	Class constraint
local count dispersed (global count) sided or both	CDG d^* (FG 2010) $+$ choices $(d_1 \vert d_2)^*$ (LACL 2011) $+$ sequences $(d_1 \bullet d_2)^*$ (LACL 2016)	(synt — sem) Simple K-star ICFI 2016, MLJ 2021 K-star revealing (on $\Delta(G)$ semantic)

- beyond d^* : repeating $\ / d_2 \ / d_1 \ / d_2 \ / d_1$, etc. as $\ / (d_1 \bullet d_2)^*$
  a proposal for extended CDG, with iterated sequences of dep.

and TGE-like algorithm for sequences of length 2 [LACL 2016]

Some open questions

from [Dekhtyar-Dikovsky-Karlov, TCS 2015]

(CDG-languages as a class of push-down automata with independent counters)

- An effective tool for showing L is not a CDG-language ?

  status of the copy language $\{ww|w \in \{a,b\}^*\}$?
- Relationships between CDG and other classes of languages ?
- Does the number of non-proj. dep. define a strict hierarchy ?
- Closure under iteration ?

  [Kanazawa Wollic (2016) "Abstract Families of Abstract Categorial Languages"

  Abstract Family of Languages (*full* AFL) if closed under
  - union ✓, concatenation ✓, Kleene plus / *Kleene star*,
  - ϵ-free homomorphism ✓/ *homomorphism*,
    inverse homomorphism ✓
  - and intersection with regular sets ✓

Control over non-projective dependencies ?

THANK YOU !