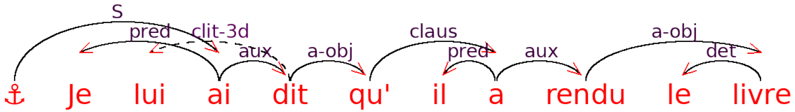
Categorial Dependency Grammars extended with typed barriers

Denis Béchet, Nantes University, France
Annie Foret, University of Rennes and IRISA, France

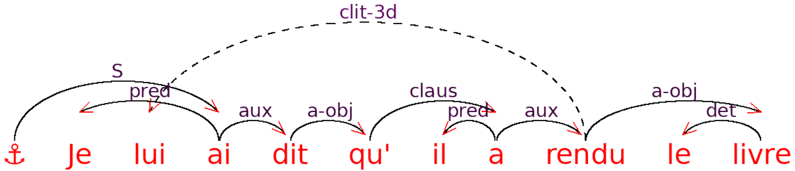
MCLP 2025, September 15–18, 2025, Orsay, France

CDG Problem 1: Overgeneration with non-projective dependencies

The CDG analyses of "Je lui ai dit qu'il a rendu le livre"
"I have told him that he has returned the book" – clitic "lui" (him)



The normal analysis



A wrong analysis

$\Longrightarrow$ Our solution: CDG$_{tb}$ (typed barriers)

CDG Problem 2: No known construction for Kleene plus

No known construction for the Kleene plus (CDG):

Let G be a CDG. $L(G)$ is the formal language generated by G

$$\exists G', \text{ a CDG such that } L(G') = L(G)^+ \text{ ?}$$

$\Longrightarrow$ Our solutions: CDG_{tb} (typed barriers) or CDG_b (barriers)

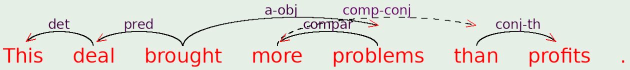
Plan

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

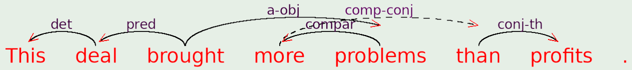
deal has $-$valency pred of a right adjacent word

Saturation of valency pred determines projective dependency

deal $\xleftarrow{\text{pred}}$ *brought* (**Governor:** *brought*, **Subordinate:** *deal*)

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*)

CDG Types express dependency valencies

PROJECTIVE DEPENDENCIES

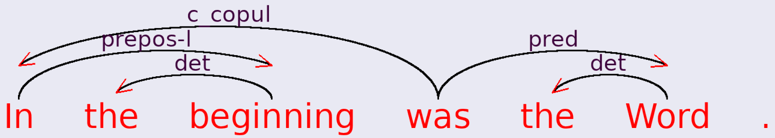
Dependency: $Gov \xrightarrow{d} Sub$:

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

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

[...] : Part of a type for projective relations (basic dependency type)
P : Part of a type for non-projective dependencies (potential)

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

[..] : Part of a type for projective relations (basic dependency type)
$..\nearrow d..$: Part of a type for non-projective dependencies (potential)

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 calculus

Left-oriented rules

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

Left-oriented rules

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

$\mathbf{L^l_\varepsilon}$. $[\;]^P[\beta]^Q \vdash [\beta]^{PQ}$ $\qquad$ (no new dependency)

CDG calculus

Left-oriented rules

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

$\mathbf{L^l_\varepsilon}$. $[\;]^P[\beta]^Q \vdash [\beta]^{PQ}$ $\qquad$ (no new dependency)

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

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

CDG calculus

Left-oriented rules

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

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

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

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

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

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 $\swarrow C, \nwarrow C$

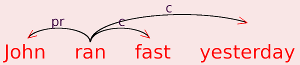
LEXICON:

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

Derivation

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

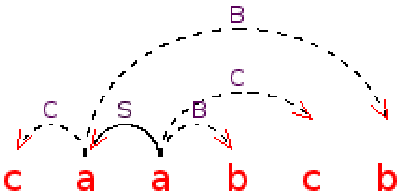
Dependency structure



$\mathbf{L^l}$	$[C]^P[C\backslash\beta]^Q \vdash [\beta]^{PQ}$	$\mathbf{L^r}$	$[\beta/C]^P[C]^Q \vdash [\beta]^{PQ}$
$\mathbf{L^l_\varepsilon}$	$[\ ]^P[\beta]^Q \vdash [\beta]^{PQ}$	$\mathbf{L^r_\varepsilon}$	$[\beta]^P[\ ]^Q \vdash [\beta]^{PQ}$
$\mathbf{I^l}$	$[C]^P[C*\backslash\beta]^Q \vdash [C*\backslash\beta]^{PQ}$	$\mathbf{I^r}$	$[\beta/C*]^P[C]^Q \vdash [\beta/C*]^{PQ}$
Ω^l	$[C*\backslash\beta]^P \vdash [\beta]^P$	Ω^r	$[\beta/C*]^P \vdash [\beta]^P$
$\mathbf{D^l}$	$\alpha^{P_1(\swarrow V)P(\nwarrow V)P_2} \vdash \alpha^{P_1PP_2}$, if FA	$\mathbf{D^r}$	$\alpha^{P_1(\nearrow V)P(\searrow V)P_2} \vdash \alpha^{P_1PP_2}$, if FA

CDG formal example: mix of n a, b and c

$$
\begin{array}{ll}
a & [S]^{\nwarrow B \nwarrow C} \\
 & [S\backslash S]^{\nwarrow B \nwarrow C} \\
 & [S]^{\nearrow B \nearrow C} \\
 & [S\backslash S]^{\nearrow B \nearrow C} \\
 & [S]^{\nwarrow B \nearrow C} \\
 & [S\backslash S]^{\nwarrow B \nearrow C} \\
 & [S]^{\nwarrow C \nearrow B} \\
 & [S\backslash S]^{\nwarrow C \nearrow B} \\
b & [\ ]^{\swarrow B} \\
 & [\ ]^{\searrow B} \\
c & [\ ]^{\swarrow C} \\
 & [\ ]^{\searrow C}
\end{array}
$$



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.

Plan

CDG example: $a^n b^n c^n$

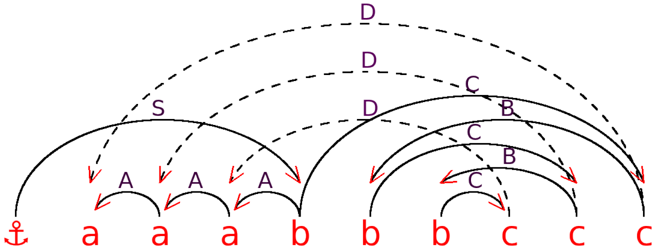
$$a \quad [A]^{\swarrow D} \qquad\qquad b \quad [A\backslash S/C] \qquad\qquad c \quad [C]^{\nwarrow D}$$
$$ \quad [A\backslash A]^{\swarrow D} \qquad\quad \quad [B/C] \qquad\qquad \quad [B\backslash C]^{\nwarrow D}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\overset{a}{A}]^{\swarrow D}\ [A\backslash \overset{a}{A}]^{\swarrow D}}{[A]^{\swarrow D\swarrow D}}\,\mathbf{L^l}\quad [A\backslash \overset{b}{S}/C]}{[S/C]^{\swarrow D\swarrow D}}\,\mathbf{L^l}\qquad \dfrac{\dfrac{[\overset{b}{B/C}]\ [\overset{c}{C}]^{\nwarrow D}}{[B]^{\nwarrow D}}\,\mathbf{L^r}\quad [B\backslash \overset{c}{C}]^{\nwarrow D}}{[C]^{\nwarrow D\nwarrow D}}\,\mathbf{L^l}}{[S]^{\swarrow D\swarrow D\nwarrow D\nwarrow D}}\,\mathbf{L^r}}{[S]^{\swarrow D\nwarrow D}}\,\mathbf{D^l}}{[S]}\,\mathbf{D^l}$$

A CDG for $\{a^n b^n c^n, n \geq 1\}$ with a derivation for *aabbcc* $(n = 2)$

CDG example: $a^n b^n c^n$

a	$[A]^{\swarrow D}$	b	$[A\backslash S/C]$	c	$[C]^{\nwarrow D}$
	$[A\backslash A]^{\swarrow D}$		$[B/C]$		$[B\backslash C]^{\nwarrow D}$



The same CDG for $\{a^n b^n c^n, n \geq 1\}$ with the dependency structure for *aaabbbccc* ($n = 3$)

Parsing time complexity : $\mathcal{O}(n^4)$

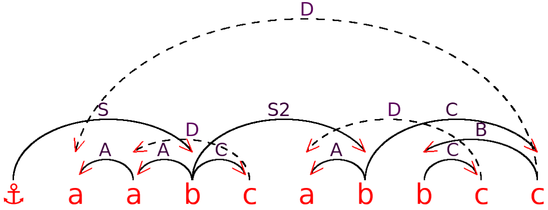
Is it possible to define a CDG that yields the product of $a^n b^n c^n$ with itself ?

$$\{a^p b^p c^p a^q b^q c^q, p \geq 1, q \geq 1\}$$

How can we find it from a CDG that yields $a^n b^n c^n$?

CDG example: An unsuccessful attempt for the product of $a^n b^n c^n$ with itself

$$\begin{array}{ll} a & [A]^{\swarrow D} \\ & [A\backslash A]^{\swarrow D} \\ b & [A\backslash S/S2/C] \\ & [A\backslash S2/C] \\ & [B/C] \\ c & [C]^{\nwarrow D} \\ & [B\backslash C]^{\nwarrow D} \end{array}$$



The CDG is built from the initial CDG for $a^n b^n c^n$:

The initial type of b with S is duplicated and $S2$ is added

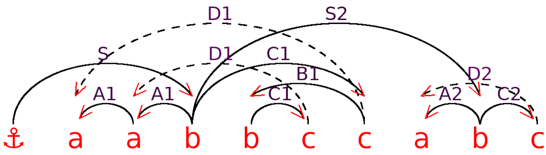
The CDG doesn't yield the product of $a^n b^n c^n$ with itself:

aabcabbcc can be parsed but it isn't correct:

Non-projective dependencies between the parts are allowed

CDG example: A correct product of $a^n b^n c^n$ with itself

$$
\begin{array}{ll}
a & [A_1]^{\swarrow D_1} \\
 & [A_1 \backslash A_1]^{\swarrow D_1} \\
b & [A_1 \backslash \color{green}{S} / \color{blue}{S_2} / C_1] \\
 & [B_1 / C_1] \\
c & [C_1]^{\nwarrow D_1} \\
 & [B_1 \backslash C_1]^{\nwarrow D_1} \\
a & [A_2]^{\swarrow D_2} \\
 & [A_2 \backslash A_2]^{\swarrow D_2} \\
b & [A_2 \backslash \color{blue}{S_2} / C_2] \\
 & [B_2 / C_2] \\
c & [C_2]^{\nwarrow D_2} \\
 & [B_2 \backslash C_2]^{\nwarrow D_2}
\end{array}
$$



All the types are duplicated from the initial CDG for $a^n b^n c^n$

$\Longrightarrow$ Two non-projective dependency names: D_1 and D_2 rather than D

$\Longrightarrow$ Higher parsing time complexity: $\mathcal{O}(n^5)$ rather than $\mathcal{O}(n^4)$

No known general construction for the Kleene plus of a CDG

Plan

Our proposal : CDG$_{tb}$ calculus with typed barriers

Left-oriented rules

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

$\mathbf{L^l_\varepsilon}$. $[\;]^P[\beta]^Q \vdash [\beta]^{PQ}$ (no new dependency)

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

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

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

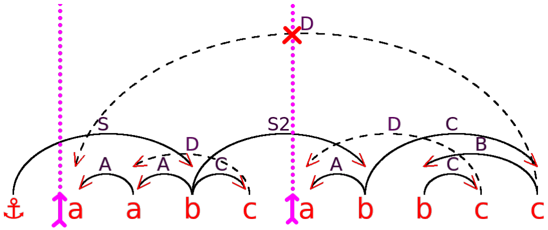
First-Available Rule (and no intermediate typed barrier)

FA$_{tb}$: 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 $\swarrow C, \nwarrow C$ and $\updownarrow C$

Potentials contain polarized valencies $\swarrow d, \nwarrow d, \searrow d, \nearrow d$ and typed barriers $\updownarrow d$

CDG_{tb} with typed barriers: a simple product of $a^n b^n c^n$ with itself

$a \quad [A]^{\updownarrow D \swarrow D}$
$\quad\;\; [A\backslash A]^{\swarrow D}$
$b \quad [A\backslash S/S2/C]$
$\quad\;\; [A\backslash S2/C]$
$\quad\;\; [B/C]^{\nwarrow D}$
$c \quad [C]^{\nwarrow D}$
$\quad\;\; [B\backslash C]^{\nwarrow D}$



There is a typed barrier $\updownarrow D$ on the rightmost a (for $aabcabbcc$)
$\Longrightarrow$ The top non-projective dependency isn't allowed this time
The CDG_{tb} with typed barriers yields the product of $a^n b^n c^n$ with itself

Only one non-projective dependency name (D)
$\Longrightarrow$ Same parsing time complexity as $a^n b^n c^n$

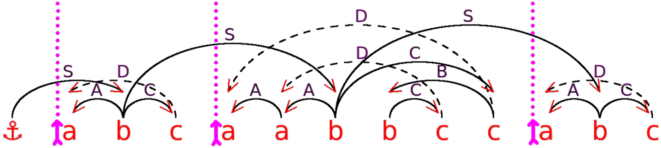
CDG$_{tb}$ with typed barriers: Kleene plus of $a^n b^n c^n$

Is it possible to define a CDG that yields Kleene plus of $a^n b^n c^n$?
$\{a^{p_1} b^{p_1} c^{p_1} a^{p_2} b^{p_2} c^{p_2} \cdots a^{p_n} b^{p_n} c^{p_n}, n \geq 1\ p_1 \geq 1, \ldots, p_n \geq 1\}$

How can we find it from a CDG that yields $a^n b^n c^n$?

No known general construction for the Kleene plus of a CDG
Always possible with a CDG_{tb} (our proposal)

$a \quad [A]^{\updownarrow D \swarrow D}$
$\quad\;\; [A\backslash A]^{\swarrow D}$
$b \quad [A\backslash S/S/C]$
$\quad\;\; [A\backslash S/C]$
$\quad\;\; [B/C]$
$c \quad [C]^{\nwarrow D}$
$\quad\;\; [B\backslash C]^{\nwarrow D}$



A typed barrier on the leftmost *a* of each part of the Kleene plus

$\Longrightarrow$ Non-projective dependencies between parts aren't allowed

$\Longrightarrow$ The CDG_{tb} yields the Kleene plus of $a^n b^n c^n$

Only one non-projective dependency name (D)

$\Longrightarrow$ Same parsing time complexity as $a^n b^n c^n$

Kleene plus: The general construction for a CDG$_{tb}$ language

Starting with G, a CDG$_{tb}$ with typed barriers

1. Transform G if G has types with empty heads in the lexicon
2. Transform G if the axiom S is used as an argument of a type
3. Transform G such that the types in the lexicon are divided in two parts :
    - The types only used on the rightmost token of any derivation
    - The types never used on the rightmost token of any derivation
4. Add (typed) barriers in the potential of the types that can only be used on the rightmost token of any derivation
5. For each type with the axiom S as head type, duplicate the same type but where S is replaced with S/S

The final CDG$_{tb}$ corresponds to the Kleene plus of the initial CDG$_{tb}$

Example: Kleene plus of $a^n b^n c^n$

$$
\begin{array}{llllll}
a & [A]^{\swarrow D} & b & [A\backslash S/C] & c & [C]^{\nwarrow D} \\
 & [A\backslash A]^{\swarrow D} & & [B/C] & & [B\backslash C]^{\nwarrow D}
\end{array}
$$

1. Transform G if G has types with empty heads in the lexicon
   $\Longrightarrow$ Ok (no empty head)
2. Transform G if the axiom S is used as an argument of a type
   $\Longrightarrow$ Ok (S only used as head type))

Example: Kleene plus of $a^n b^n c^n$

a	$[A]^{\swarrow D}$	b	$[A\backslash S/C]$	c	$[C]^{\nwarrow D}$
	$[A\backslash A]^{\swarrow D}$		$[B/C]$		$[B\backslash C]^{\nwarrow D}$

3. Transform G such that the types in the lexicon are divided in two parts :
   - The types only used on the rightmost token of any derivation
   - The types never used on the rightmost token of any derivation

   Types on the rightmost token: Types of c ($[C]^{\nwarrow D}$ and $[B\backslash C]^{\nwarrow D}$)

   Types on other tokens: All the types

   Not ok (the types of c)

   $\Longrightarrow$ We need to transform the grammar (axiom S_r):

a	$[A_r]^{\swarrow D}$	b	$[A_o\backslash S_r/C_r]$	c	$[C_r]^{\nwarrow D}$
	$[A_o]^{\swarrow D}$		$[A_o\backslash S_o/C_o]$		$[C_o]^{\nwarrow D}$
	$[A_o\backslash A_r]^{\swarrow D}$		$[B_r/C_r]$		$[B_o\backslash C_r]^{\nwarrow D}$
	$[A_o\backslash A_o]^{\swarrow D}$		$[B_o/C_o]$		$[B_o\backslash C_o]^{\nwarrow D}$

Remark: The grammar can be simplified (useless types)

$a \quad [A_o]^{\swarrow D}$, $[A_o\backslash A_o]^{\swarrow D}$

$b \quad [A_o\backslash S_r/C_r]$, $[B_o/C_o]$

$c \quad [C_r]^{\nwarrow D}$, $[C_o]^{\nwarrow D}$, $[B_o\backslash C_r]^{\nwarrow D}$, $[B_o\backslash C_o]^{\nwarrow D}$

4. Add typed barriers in the potential of the types that can only be used on the rightmost token of any derivation

$a \quad [A_o]^{\swarrow D}$, $[A_o\backslash A_o]^{\swarrow D}$

$b \quad [A_o\backslash S_r/C_r]$, $[B_o/C_o]$

$c \quad [C_r]^{\nwarrow D \uparrow D}$, $[C_o]^{\nwarrow D}$, $[B_o\backslash C_r]^{\nwarrow D \uparrow D}$, $[B_o\backslash C_o]^{\nwarrow D}$

$a \quad [A_o]^{\swarrow D}$
$[A_o \backslash A_o]^{\swarrow D}$

$b \quad [A_o \backslash S_r / C_r]$
$[B_o / C_o]$

$c \quad [C_r]^{\nwarrow D \updownarrow D}$
$[C_o]^{\nwarrow D}$
$[B_o \backslash C_r]^{\nwarrow D \updownarrow D}$
$[B_o \backslash C_o]^{\nwarrow D}$

5. For each type with the axiom S_r as head type, duplicate the same type but where S_r is replaced with S_r / S_r

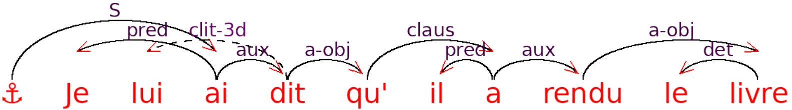
$a \quad [A_o]^{\swarrow D}$
$[A_o \backslash A_o]^{\swarrow D}$

$b \quad [A_o \backslash S_r / C_r]$
$[A_o \backslash S_r / S_r / C_r]$
$[B_o / C_o]$

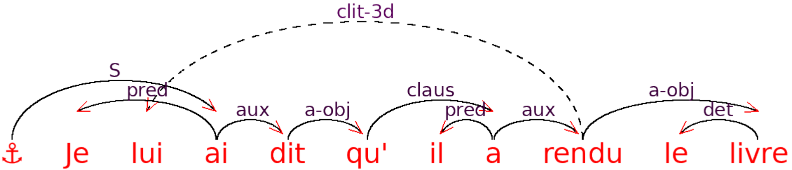
$c \quad [C_r]^{\nwarrow D \updownarrow D}$
$[C_o]^{\nwarrow D}$
$[B_o \backslash C_r]^{\nwarrow D \updownarrow D}$
$[B_o \backslash C_o]^{\nwarrow D}$

Plan

The CDG analyses of “Je lui ai dit qu’il a rendu le livre”
“I have told him that he has returned the book” clitic “lui” (him)
$il \mapsto [\;]^{\swarrow clit-3d}$ $dit, rendu \mapsto [aux/a-obj], [aux/a-obj]^{\nwarrow clit-3d}$



The normal analysis: $dit \mapsto [aux/a-obj]^{\nwarrow clit-3d}$ $rendu \mapsto [aux/a-obj]$

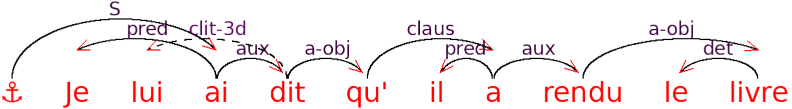


A wrong analysis: $dit \mapsto [aux/a-obj]$ $rendu \mapsto [aux/a-obj]^{\nwarrow clit-3d}$

CDG and CDG_{tb} for Natural Languages

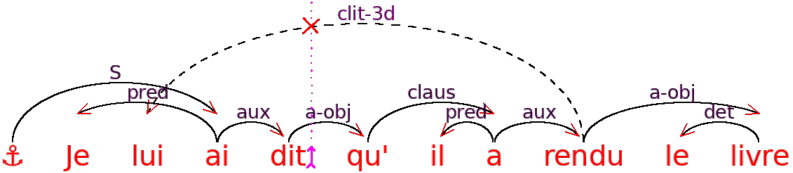
The CDG_{tb} analyses of “Je lui ai dit qu’il a rendu le livre”
“I have told him that he has returned the book” clitic “lui” (him)

$il \mapsto [\ ]^{\swarrow clit-3d}$ $dit, rendu \mapsto [aux/a-obj]^{\updownarrow clit-3d}, [aux/a-obj]^{\nwarrow clit-3d}$



The normal analysis: $dit \mapsto [aux/a-obj]^{\nwarrow clit-3d}$

$rendu \mapsto [aux/a-obj]^{\updownarrow clit-3d}$



No analysis: $dit \mapsto [aux/a-obj]^{\updownarrow clit-3d}$ $rendu \mapsto [aux/a-obj]^{\nwarrow clit-3d}$

$\Longrightarrow$ Typed barriers can control the range of specific non-projective dependencies

Conclusion and Open Questions

- The product and the Kleene plus of languages may be useful, for instance, to model the conjunction of parts of speech or the list of complex complements in a lot of natural languages.
- Our proposal allows such constructions for CDG_{tb} languages (with typed barriers).
- There is no parsing complexity penalty for the product and the Kleene plus.
- Categorial Dependency Grammars extended with typed barriers define an Abstract Family of Languages (closed under union, product, Kleene plus, ε-free homomorphism, inverse homomorphism, intersection with regular sets).
- The same questions remain opened for classical CDG.
- For natural languages, typed barriers can control the range of specific non-projective dependencies.

Conclusion and Open Questions

- The product and the Kleene plus of languages may be useful, for instance, to model the conjunction of parts of speech or the list of complex complements in a lot of natural languages.
- Our proposal allows such constructions for CDG_{tb} languages (with typed barriers).
- There is no parsing complexity penalty for the product and the Kleene plus.
- Categorial Dependency Grammars extended with typed barriers define an Abstract Family of Languages (closed under union, product, Kleene plus, ε-free homomorphism, inverse homomorphism, intersection with regular sets).
- The same questions remain opened for classical CDG.
- For natural languages, typed barriers can control the range of specific non-projective dependencies.

THANK YOU !

Bibliography

Michael Dekhtyar, Alexander Dikovsky, and Boris Karlov.
Categorial dependency grammars.
Theoretical Computer Science, 579:33–63, 2015.

Y. Bar-Hillel, H. Gaifman, and E. Shamir.
On categorial and phrase structure grammars.
Bull. Res. Council Israel, 9F:1–16, 1960.

I. Mel'čuk.
Dependency Syntax.
SUNY Press, Albany, NY, 1988.

Denis Béchet and Annie Foret.
Categorial dependency grammars extended with barriers (CDGb) yield an abstract family of languages (AFL).
In *David C. Wyld and Dhinaharan Nagamalai, editors, Proceedings of the 5th International Conference on Natural Language Processing and Computational Linguistics (NLPCL 2024), Copenhagen, Denmark, pages 53–66*, September