Categorial Dependency Grammars extended with barriers (CDG_b) yield an Abstract Family of Languages (AFL)

Denis Béchet[1] and Annie Foret[2]

[1] Nantes University, France
[2] Univ. Rennes and IRISA, France

Abstract

We consider the family of Categorial Dependency Grammars (CDG), as computational grammars for language processing. CDG are a class of categorial grammars defining dependency structures. They can be viewed as a formal system, where types are attached to words, combining the classical categorial grammars' elimination rules with valency pairing rules that are able to define non-projective (discontinuous) dependencies.
Whereas the closure problem under iteration is open for the original version of CDG, we define "CDG extended with barriers" as an extended version of the original CDG that solves this formal issue. We provide a rule system and show that the extended version defines an Abstract Family of Languages (AFL) while preserving advantages of the original CDG in terms of expressivity, parsing and efficiency.

Keywords

Logical approach to natural language, Type calculus, Categorial Grammar, Dependency Grammar, Abstract Family of Languages

1. Introduction

Categorial Dependency Grammars (CDG) [1] are a class of categorial grammars [2] that define dependency structures [3]. CDG are a unique class of grammars directly generating unbounded dependency structures (DS), beyond context-freeness, able to define non-projective dependency structures, but remain well adapted to real NLP applications.

CDG can be viewed as a formal system, where types are attached to words, combining the classical categorial grammars' elimination rules with valency pairing rules that are able to define non-projective (discontinuous) dependencies. An overview of this class is provided in [4].

Some closure properties have been shown in [1] for the class of string-languages generated by CDG (union, etc.), but some closure questions remain open. In particular, we do not know whether the class of string-languages generated by CDG is closed for Kleene plus (the conjecture is "no" in [1]) and whether they are an Abstract Family of Languages (AFL).

AFL closure properties are nice properties expected for standard grammar classes and have been shown for several grammatical frameworks: we refer in particular to [5] for multiple context-free grammars. The AFL properties are also nice as they allow a meta-level modular construction of grammars.

In this paper, we define "CDG extended with barriers" (CDG_b), an extended version of the original CDG. Our contribution is to propose this extended version CDG_b and to show that it

David C. Wyld et al. (Eds): CCSIT, NLPCL, AISC, ITE, NCWMC, DaKM, BIGML, SIPP, SOEN, PDCTA – 2024
pp. 53-66, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141706

defines an Abstract Family of Languages (AFL), while preserving advantages of the original CDG, in terms of expressivity, parsing and efficiency. For natural langage modelling, this new version allows to block some unwanted word links.

The rest of the paper is organized as follows: Section 2 contains background knowledge, we give preliminaries on the notion of Abstract Family of Languages (AFL) ; Section 3 contains our method, we introduce CDG extended with barriers (CDG$_b$); Sections 4 and 5 contain the main results ; in Section 4, we provide technical properties on CDG$_b$, related to grammar or derivation equivalences, that are helpful for closure properties; in Section 5, we establish the closure properties constituting an AFL. Section 6 concludes.

2. PRELIMINARIES ON ABSTRACT FAMILY OF LANGUAGES (AFL)

We are interested in closure properties of a family $\mathcal{F}$ of languages, as those of AFL. Before considering such questions for CDG, we give background definitions [6].

Homomorphisms. For finite alphabets V_1, V_2: a homomorphism[1] h from V_1^* to V_2^* is ϵ-free if $h(w) = \epsilon$ implies $w = \epsilon$.

A family $\mathcal{F}$ is closed under inverse homomorphism if whenever $L \subseteq V_1^*$ is in $\mathcal{F}$ and h is a homomorphism from V_2^* to V_1^*, then $h^{-1}(L)$ is also in $\mathcal{F}$, where:

$$h^{-1}(L) = \{w \in V_2^* \mid h(w) \in L\}$$

Substitutions. A substitution is a mapping f from V_1 to $\mathcal{P}(V_2^*)$, it is naturally extended to strings in V_1^* (by concatenation) and to sets of strings (by union)[2].

A family $\mathcal{F}$ is closed under substitution if whenever $L \in V_1^*$ is in $\mathcal{F}$ and f is a substitution from V_1 such that $f(a) \in \mathcal{F}$ for all $a \in V_1$, then $f(L)$ is also in $\mathcal{F}$.

AFL. $\mathcal{F}$ is an Abstract Family of Languages (AFL) if it is closed under union, concatenation, Kleene plus, ϵ-free homomorphism, inverse homomorphism and intersection with regular sets. A *full AFL* is defined similarly, with Kleene star (not just Kleene plus) and arbitrary homomorphisms (not just ϵ-free).

For example, the class of multiple context-free grammars yields an AFL [7]. Simpler classes such as regular languages and context-free languages are AFL too. The class of string languages generated by abstract categorial grammars is a substitution-closed full AFL, as shown in [6].

CDG-languages are closed under the following AFL-operations: union, concatenation, ϵ-free homomorphism, inverse homomorphism and intersection with regular sets. The CDG family is thus a trio (closed under ϵ-free homomorphism, inverse homomorphism, and intersection with regular language) and also a semi-AFL (a trio closed under union). However the AFL question is open for CDG-languages as we do not know if they are closed for Kleene plus (the conjecture is "no" in [1]). In [8] it is shown that the mmCDG class [9], extending CDG with a multimodal rule, defines an AFL.

Note that these closure properties are established for string-languages. Some other works consider structure-languages: in [6] closure properties for ACG tree-languages are also shown. In the case of CDG, such closure questions could be addressed at the level of dependency structures too. This paper provides closure properties for string-languages.

3. CDG EXTENDED WITH BARRIERS: CDG$_b$

As for other categorial grammars, a CDG or a CDG$_b$ defines a lexicon and the rules used in the calculus are fixed. The lexicon maps each word or symbol to one or several types. For instance,

[1] each character is replaced by a single string, with $h(uv) = h(u)h(v)$ and $h(\epsilon) = \epsilon$

[2] $f(\epsilon) = \{\epsilon\}$, $f(ws) = f(w)f(s)$, $f(\{w\}) = f(w)$, $f(\bigcup_i L_i) = \bigcup_i f(L_i)$

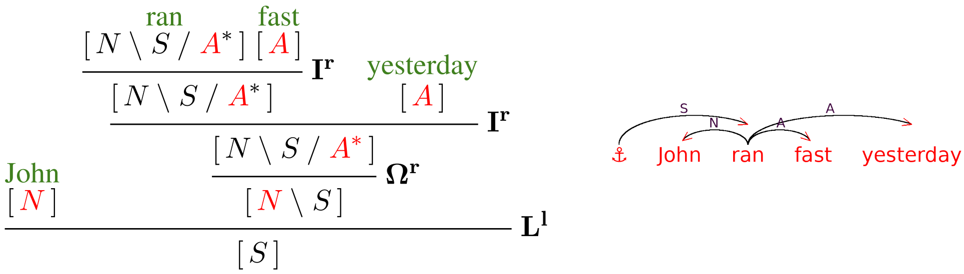


Figure 1. A derivation (on the left) and its dependency structure (on the right)

the following CDG lexicon gives a unique type to the words John, ran, fast and yesterday:

John $\mapsto$ $[N]$ ran $\mapsto$ $[N\backslash S/A^*]$ fast, yesterday $\mapsto$ $[A]$

where types are built from the following primitive types: A for adverbs, N for nouns, and S for sentences, using categorial $/$ $\backslash$ operators in Lambek notation. Moreover, the type of ran has an iterated dependency type A^* that can introduce several projective dependencies A with the same governor ran. The string John ran fast yesterday is recognized by the CDG. A proof is given by the derivation in Figure 1. In this derivation, the words are written just above the type that has been chosen for it in the lexicon. The derivation ends by the axiom S. Each node corresponds to the application of one of the rules of the calculus of dependency types. Most rules in a derivation create a new dependency in the dependency structure as examplified in Figure 1. This example involves only basic rules, for more complex constructs see next sections.

3.1. CDG$_b$: types, proofs and derivations

CDG$_b$ are extensions of CDG where barriers $\updownarrow$ may be added to types. Some rules are modified in CDG$_b$ to take into account the barriers in potentials.

Definition 1 (CDG$_b$ Types). *Let* $\mathbf{C}$ *be a set of* local dependency names [3] *and* $\mathbf{V}$ *be a set of* valency names. *A* CDG$_b$ dependency type *is an expression* B^P *in which* B *is a basic dependency type and* P *is a b-potential, using next definitions.* $\mathbf{CAT}_b(\mathbf{C},\mathbf{V})$ *will denote the set of all CDG$_b$ dependency types over* $\mathbf{C}$ *and* $\mathbf{V}$.

1. *An expression of the form* d^* *where* $d \in \mathbf{C}$, *is called an* iterated dependency type.

2. *Local dependency names and iterated dependency types are* primitive types.

3. *An expression of the form*

   $t = [l_m \backslash \ldots \backslash l_1 \backslash H / r_1 / \ldots / r_n]$
   in which $m, n \geq 0$, $l_1, \ldots, l_m, r_1, \ldots, r_n$ *are primitive types*
   and H *is either a local dependency name (in* $\mathbf{C}$*) or is empty (written* ε*),*

   is called a basic dependency type*;*

   $l_1, \ldots, l_m$ *and* $r_1, \ldots, r_n$ *are left and right* argument types *of* t*;*

   H *is called the* head type *of* t.

4. *The expressions of the form* $\swarrow v$, $\nwarrow v$, $\searrow v$, $\nearrow v$, *where* $v \in \mathbf{V}$, *are called* polarized valencies, *with characteristics as follows:*

[3] called elementary (dependency) categories in [1] ; several terminologies have been used, the version in this article does not use the term anchors but they are seen as particular local dependencies

polarized valency	*polarity*	*arrow direction*	*dual*	*left / right bracket*
$\nearrow v$	*positive*	*left-to-right*	$\searrow v$	*left*
$\searrow v$	*negative*	*left-to-right*	$\nearrow v$	*right*
$\swarrow v$	*negative*	*right-to-left*	$\nwarrow v$	*left*
$\nwarrow v$	*positive*	*right-to-left*	$\swarrow v$	*right*

5. *A (possibly empty) string P of polarized valencies is called a* potential.

6. *The expressions of the form $\swarrow v$, $\nwarrow v$, $\searrow v$, $\nearrow v$ and $\updownarrow$, where $v \in \mathbf{V}$, are called* b-extended polarized valencies.

7. *A (possibly empty) string P of b-extended polarized valencies is called a* b-potential.

Basic dependency types can also be viewed as CDG$_b$ types with empty b-potential, as $[N \backslash S / A^*]$ in Figure 1. In this figure, the primitive types A, N, S are local dependency names and $A*$ is both an iterated dependency type and a primitive type. Section 3.2 provides examples with potentials generating another kind of dependencies displayed as dashed arrows in the dependency structures.

Restriction to CDG. The difference between a CDG type and a CDG$_b$ type lies in the polarity part, where barriers are not allowed in CDG.

A *dependency type* (CDG Type) is an expression B^P in which B is a basic dependency type and P is a potential. $\mathbf{CAT}(\mathbf{C}, \mathbf{V})$ will denote the set of all dependency types over $\mathbf{C}$ and $\mathbf{V}$.

Local Dependency names, iterated dependency types, *primitive types* are defined for CDG$_b$ as for CDG, as well as *basic dependency types* and their *argument types* and *head types*. *Polarized valencies* are defined for CDG$_b$ as for CDG, but we now add barriers $\updownarrow$.

Definition 2 (Set of rules). *In this set of rules on lists of types, the symbol C stands for a local dependency name. The symbol α is a basic dependency type. The symbol β ranges over expressions of the form $l_m \backslash \ldots \backslash l_1 \backslash H / r_1 / \ldots / r_n$*

$\mathbf{L^l}$	$[C]^P[C \backslash \beta]^Q \vdash [\beta]^{PQ}$	$\mathbf{L^r}$	$[\beta / C]^P[C]^Q \vdash [\beta]^{PQ}$
$\mathbf{L^l_\varepsilon}$	$[\varepsilon]^P[\beta]^Q \vdash [\beta]^{PQ}$	$\mathbf{L^r_\varepsilon}$	$[\beta]^P[\varepsilon]^Q \vdash [\beta]^{PQ}$
$\mathbf{I^l}$	$[C]^P[C^* \backslash \beta]^Q \vdash [C^* \backslash \beta]^{PQ}$	$\mathbf{I^r}$	$[\beta / C^*]^P[C]^Q \vdash [\beta / C^*]^{PQ}$
$\mathbf{\Omega^l}$	$[C^* \backslash \beta]^P \vdash [\beta]^P$	$\mathbf{\Omega^r}$	$[\beta / C^*]^P \vdash [\beta]^P$
$\mathbf{D^l}$	$\alpha^{P_1 \swarrow v P \nwarrow v P_2} \vdash \alpha^{P_1 P P_2}$	$\mathbf{D^r}$	$\alpha^{P_1 \nearrow v P \searrow v P_2} \vdash \alpha^{P_1 P P_2}$

In $\mathbf{D^l}$, the potential $P_1 \swarrow v P \nwarrow v P_2$ satisfies the pairing rule $\mathbf{FA}_b$:

$\mathbf{FA}_b$ (*First Available between barriers*): P has no occurrence of $\swarrow v$ or $\nwarrow v$ **and no barrier**.

In $\mathbf{D^r}$, the potential $P_1 \nearrow v P \searrow v P_2$ satisfies the pairing rule $\mathbf{FA}_b$:

$\mathbf{FA}_b$ (*First Available between barriers*): P has no occurrence of $\nearrow v$ or $\searrow v$ **and no barrier**.

The calculus defines the immediate provability relation $\vdash_b$ on strings of CDG$_b$ types. Its transitive closure $\vdash_b^*$ defines a derivation when the right part is reduced to a type $[S]^{\updownarrow \updownarrow \cdots \updownarrow}$ where S is an axiom.

In the CDG case. The set of rules is the same, the original pairing rules are:

In $\mathbf{D^l}$, the potential $P_1 \swarrow v P \nwarrow v P_2$ satisfies the pairing rule $\mathbf{FA}$:

$\mathbf{FA}$ (*First Available*): P has no occurrence of $\swarrow v$ or $\nwarrow v$.

In $\mathbf{D^r}$, the potential $P_1 \nearrow v P \searrow v P_2$ satisfies the pairing rule $\mathbf{FA}$:

$\mathbf{FA}$ (*First Available*): P has no occurrence of $\nearrow v$ or $\searrow v$.

The CDG pairing rules eliminate dual dependencies. In fact, we may use the new pairing rules for CDG, with the same effect (as CDG involves no barrier).

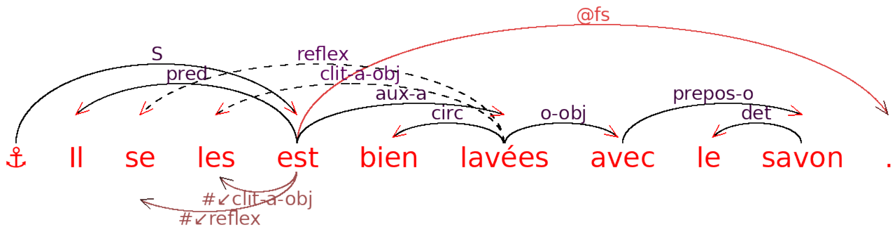


Figure 2. An example in French meaning “he washed them well with the soap”

Definition 3 (CDG$_b$ grammar and language). *A categorial dependency grammar extended with* barriers *(CDG$_b$) is a system $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$, where W is a finite set of words, $\mathbf{C}$ is a finite set of local dependency names containing the selected name S (an axiom), $\mathbf{V}$ is a finite set of valency names, and λ, called lexicon, is a finite substitution such that $\lambda(a) \subset \mathbf{CAT}_b(\mathbf{C}, \mathbf{V})$ for each word $a \in W$.*
A string $x = w_1 w_2 \cdots w_n \in W^*$ is generated by *G iff there exists a proof $\Gamma \vdash_b^* [S]^P$ where $\Gamma \in \lambda(x) = \lambda(w_1) \cdots \lambda(w_2) \cdots \lambda(w_n)$ and $P = \Uparrow\Uparrow \cdots \Uparrow$ (P is empty or contains only barriers).*
The language $L(G)$ is the set of strings of W^* that are generated by *G. $\mathcal{L}(CDG_b)$ will denote the family of languages generated by these grammars.*

A CDG is also a CDG$_b$ that defines the same string-language.

3.2. Expressive power of CDG

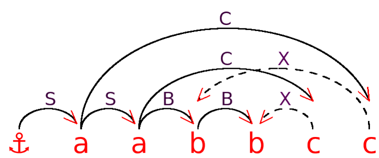
Example 1. CDG are used to model complex sentences of a natural language where dependencies are usually projective but may be sometimes non-projective, see Figure 2. Normal projective dependencies appear as black plain arrows (there is an arrow between an anchor and the main word of the sentence *est* which denotes the root node). Black dashed arrows represent non-projective dependencies that can cross other dependencies. The “host” projective dependencies *#$\swarrow$clit-a-obj* and *#$\swarrow$reflex* below the diagram are complementary to the non-projective dependencies. They fix the position of the dependent of a non-projective dependency. A red arrow *@fs* above the diagram introduces a “punctuation” projective dependency. Each projective dependency (normal, punctuation or host) corresponds to a step of one of the rules $\mathbf{L^r}$, $\mathbf{L^l}$, $\mathbf{I^r}$ and $\mathbf{I^l}$ in a derivation. The non-projective dependencies correspond to rules $\mathbf{D^r}$ and $\mathbf{D^l}$.

Example 2. Let $G_1 = (\{a, b, c\}, \{S, B, C\}, \{X\}, S, \lambda_1)$ where λ_1 is defined by:

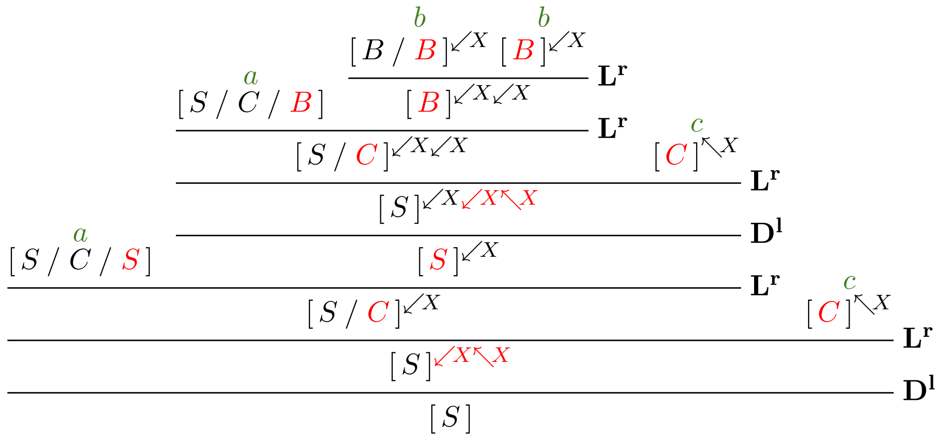
$$a \mapsto [S / C / S] \quad [S / C / B] \qquad b \mapsto [B / B]^{\swarrow X} \quad [B]^{\swarrow X} \qquad c \mapsto [C]^{\nwarrow X}$$

G_1 generates the language $L_1 = \{a^n b^n c^n \mid n > 0\}$.

For instance, G_1 generates $aabbcc$. The dependency structure is as follows:



The derivation is as follows:



For formal languages, we only use one kind of projective dependencies (corresponding to rules $\mathbf{L^r}$, $\mathbf{L^l}$, $\mathbf{I^r}$ and $\mathbf{I^l}$) and non-projective dependencies (corresponding to rules $\mathbf{D^r}$ and $\mathbf{D^l}$). There isn't any distinction between normal, "punctuation" and "host" dependencies. A special arrow that starts from an anchor marks the root node of the structure (it is not a real dependency). In a dependency structure, rules $\mathbf{L^r_\varepsilon}$, $\mathbf{L^l_\varepsilon}$, $\mathbf{\Omega^r}$ and $\mathbf{\Omega^l}$ don't introduce a dependency.

Example 3. The grammar G_1 (Example 2) viewed as a CDG$_b$ defines obviously the same language $L_1 = \{a^n b^n c^n \mid n > 0\}$. We don't know if the language L_1^+ can be generated by a CDG but it is possible with barriers to define a CDG$_b$ for it. As it is shown later, it is possible to transform G_1 into a CDG$_b$ that generates L_1^+. We start by transforming G_1 into G_2 which has an "independent main category" (see Lemma 1) then into G_3 that has a barrier on the rightmost types of a derivation (see Theorem 3) and into G_4 using the construction for Kleene plus (see Theorem 6).

Let $G_2 = (\{a, b, c\}, \{S, A, B, C\}, \{X\}, S, \lambda_2)$ where λ_2 is defined by:

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

G_2 is equivalent to G_1 but the axiom S is not used as argument of a type. G_2 is also a CDG.

Then, let $G_3 = (\{a, b, c\}, \{S, A, B, C, C'\}, \{X\}, S, \lambda_3)$ where λ_3 is defined by:

$$a \mapsto [S / C' / A]\ [A / C / A] \quad [S / C' / B]\ [A / C / B] \qquad b \mapsto [B / B]^{\swarrow X} \quad [B]^{\swarrow X} \qquad c \mapsto [C]^{\nwarrow X} \quad [C']^{\nwarrow X \updownarrow}$$

G_3 is equivalent to G_2 but, in a derivation, the rightmost type is always a type with a barrier (it is the type $[C']^{\nwarrow X \updownarrow}$).

Finally, let $G_4 = (\{a, b, c\}, \{S, A, B, C, C'\}, \{X\}, S, \lambda_4)$ where λ_4 is defined by:

$$a \mapsto [S / S / C' / A]\ [S / C' / A]\ [A / C / A] \quad [S / S / C' / B]\ [S / C' / B]\ [A / C / B]$$

$$b \mapsto [B / B]^{\swarrow X} \quad [B]^{\swarrow X} \qquad c \mapsto [C]^{\nwarrow X} \quad [C']^{\nwarrow X \updownarrow}$$

Note that λ_4 is obtained from λ_3 by adding the types where S is replaced by S / S.

G_4 generates the language L_1^+. For instance, Figure 3 shows a derivation of $abcaabbcc$.

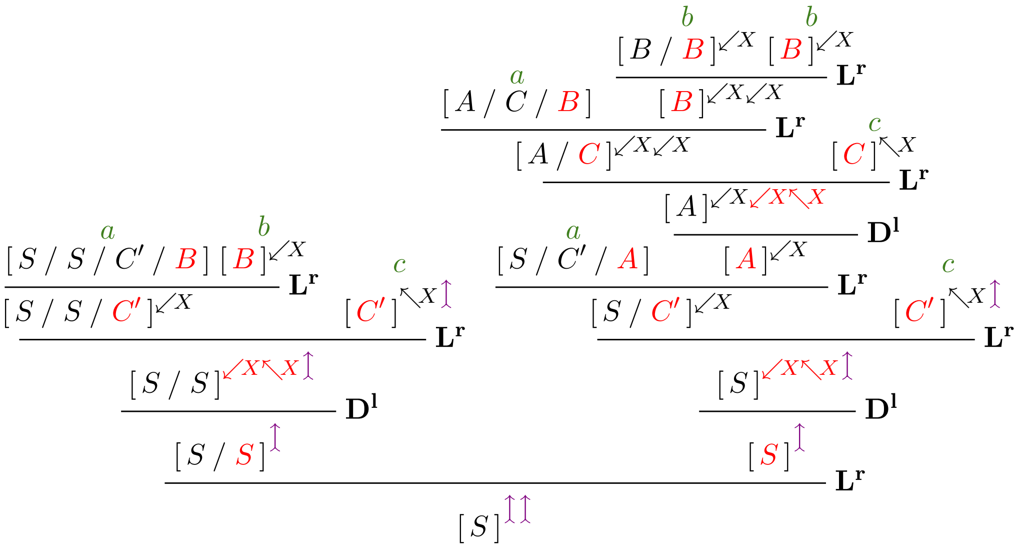


Figure 3. A derivation for $abcaabbcc$ using G_4

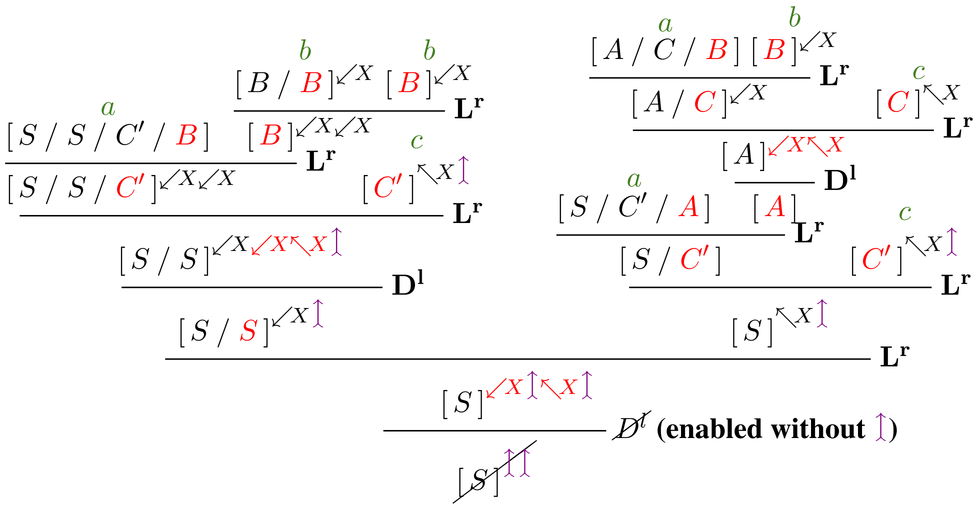


Figure 4. A forbidden proof for $abbcaabcc$ using G_4, enabled if barriers are dropped

The barrier on type $[C']^{\nwarrow X \updownarrow}$ for c in G_4 is important to limit non-projective dependencies between b and c: If the type for c is $[C']^{\nwarrow X}$ rather than $[C']^{\nwarrow X \updownarrow}$, it is possible to generate words that aren't in L_1^+. For instance $abbcaabcc$ has a derivation where the rightmost c is linked to the leftmost b. In the derivation of Figure 4, the last step using $\mathbf{D^l}$ is possible if the barriers don't exist but the derivationisn't correct if the barriers are present.

3.3. Subclasses of CDG$_b$

We define the subclasses of CDG$_b$ where the number of valency names is bound. Let $\mathcal{L}_k(CDG_b)$ denote the subclass of $\mathcal{L}(CDG_b)$ where the number of different valency names is bound by k. For a CDG$_b$ $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$, it means that $|\mathbf{V}| \leq k$.

Our results show that each subclass $\mathcal{L}_k(CDG_b)$, for each k, also defines an AFL.

4. Technical properties

In this section, we provide technical properties on CDG$_b$, related to grammar or derivation equivalences, that are helpful for closure properties.

4.1. Balancable and balanced potentials of CDG$_b$

Some definitions of [1] need modifications in order to take into account the addition of barriers. [1] shows that in a derivation, projective rules (on basic type) and non-projective rules (on polarized valencies) are independent. This property is also true for CDG$_b$. It means that the calculus can be done independently on the local projection and the valency projection of a string of types.

Definition 4. *The local projection $\|\gamma\|_l$ of a string of dependency types γ is defined as follows: $\|\varepsilon\|_l = \varepsilon$ and $\|C^P\,\alpha\|_l = C\,\|\alpha\|_l$*
The valency projection $\|\gamma\|_v$ of a string of dependency types γ is defined as follows: $\|\varepsilon\|_v = \varepsilon$ and $\|C^P\,\alpha\|_v = P\,\|\alpha\|_v$

For instance $\|[B \setminus C]^{\searrow A}\|_l = [B \setminus C]$ and $\|[B \setminus C]^{\searrow A}\|_v = \searrow A$. With CDG$_b$, the valency projection in a derivation must satisfy a new "well-bracketing" criterion. In fact, barriers divide a valency projection in sub-parts that contain no barrier and must satisfy the original "well-bracketing" defined in [1].

Definition 5. *A left bracket valency is a valency of the form $\swarrow v$ or $\nearrow v$, a right bracket valency is a valency of the form $\nwarrow v$ or $\searrow v$. For a polarized valency v and a potential P, $|P|_v$ denotes the number of occurrences of v in P.*
For a potential P, a left bracket valency v and its dual right bracket valency v':
$\Delta_v(P) = max\{|P'|_v - |P'|_{v'} \;:\; P' \text{ is a suffix of } P \text{ and } P' \text{ has no barrier}\}$
$\Delta_{v'}(P) = max\{|P'|_{v'} - |P'|_v \;:\; P' \text{ is a prefix of } P \text{ and } P' \text{ has no barrier}\}$

For instance, $\Delta_{\swarrow A}(\swarrow B \swarrow A \updownarrow \nwarrow A \swarrow A \swarrow B \swarrow A \nwarrow A \swarrow A) = 2$ and $\Delta_{\nwarrow A}(\swarrow B \swarrow A \updownarrow \nwarrow A \swarrow A \swarrow B \swarrow A \nwarrow A \swarrow A) = 0$. For a left bracket valency like $\swarrow A$, we only look at the suffixes of the right part of the potential that contains no bracket $\nwarrow A \swarrow A \swarrow B \swarrow A \nwarrow A \swarrow A$. For a right bracket valency like $\nwarrow A$, we only look at the prefixes of the left part or the potential that contains no bracket $\swarrow B \swarrow A$. These numbers are always positive or zero (ε is a prefix or a suffix).

Some potentials cannot appear in a derivation because some part ot it (delimited by barriers) does not verify a "well-bracketing" criterion. For instance, $\swarrow B \swarrow A \updownarrow \nwarrow A \swarrow A \swarrow B \swarrow A \nwarrow A \swarrow A$ can never appear in a derivation that ends with only barriers because the right part after the barrier $\nwarrow A \swarrow A \swarrow B \swarrow A \nwarrow A \swarrow A$ starts with $\nwarrow A$ whose reduction is blocked by the barrier. More generally, in a potential $P_1 \updownarrow P_2$ that contains a barrier, the left part P_1 mustn't have "pending" left bracket valencies: $\Delta_v(P_1)$ must be zero for any left bracket v. Similarly, $\Delta_{v'}(P_2)$ must be zero for any right bracket v'.

Definition 6. *A potential P is* balancable *iff for every partition of $P = P_1 \updownarrow P_2$, for every left bracket valency v and for every right bracket valency v', $\Delta_v(P_1) = \Delta_{v'}(P_2) = 0$. A potential P is* balanced *iff it is balancable and for every valency v (left or right bracket), $\Delta_v(P) = 0$.*

Theorem 1. *Let $\vdash_l^*$ be the CDG$_b$ restricted to the local rules* $\mathbf{L^l}, \mathbf{L^r}, \mathbf{L^l_\varepsilon}, \mathbf{L^r_\varepsilon}, \mathbf{I^l}, \mathbf{I^r}, \mathbf{\Omega^l}, \mathbf{\Omega^r}$. *Let $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$ be a CDG$_b$. $x \in L(G)$ iff there is a string of categories $\gamma \in \lambda(x)$ such that $\|\gamma\|_l \vdash_l^* S$ and $\|\gamma\|_v$ is balanced.*

Proof. The theorem and its proof are similar to the original ones for CDG (see Theorem 1 in [1]). This amounts to postpone rules $\mathbf{D^l}$ and $\mathbf{D^r}$. □

4.2. CDG$_b$ without empty head

The CDG and CDG$_b$ calculus enable the use of types with an empty head like $[\, A \setminus \varepsilon \,/\, B \,/\, C\,]^{\swarrow B}$. The rules $\mathbf{L}^{\mathbf{l}}_{\varepsilon}$ and $\mathbf{L}^{\mathbf{r}}_{\varepsilon}$ that are close to $\mathbf{L}^{\mathbf{l}}$ and $\mathbf{L}^{\mathbf{r}}$ can cancel such types in the presence of another type (which is not transformed). However, these types are not essential. In fact, it is possible to transform a grammar using types with an "empty head" into an equivalent grammar without such types by replacing the rules $\mathbf{L}^{\mathbf{l}}_{\varepsilon}$ and $\mathbf{L}^{\mathbf{r}}_{\varepsilon}$ by other local rules applied to a specific head type that mimics an empty head.

Theorem 2. *Let $\vdash^*_H$ be the CDG$_b$ calculus restricted to rules $\mathbf{L}^{\mathbf{l}}, \mathbf{L}^{\mathbf{r}}, \mathbf{I}^{\mathbf{l}}, \mathbf{I}^{\mathbf{r}}, \mathbf{\Omega}^{\mathbf{l}}$, $\mathbf{\Omega}^{\mathbf{r}}$, $\mathbf{D}^{\mathbf{l}}$, $\mathbf{D}^{\mathbf{r}}$ (the rules with no empty head). Let $G \in \mathcal{L}_k(CDG_b)$. There exists an equivalent grammar in $\mathcal{L}_k(CDG_b)$ using only types without empty head where proofs are based on $\vdash^*_H$ rather that $\vdash^*_b$.*

Proof. See Annex A.1. □

4.3. CDG$_b$ with a barrier on the rightmost type

Barriers may be used to prevent using $\mathbf{D}^{\mathbf{l}}$ and $\mathbf{D}^{\mathbf{r}}$ rules in a CDG$_b$ language. This is useful for the concatenation of two languages or the Kleene plus of a single language. Because CDG$_b$ are lexicalized, these barriers are added on the right[4] of the potential of certain types of the lexicon. This is correct only when we can be sure that the modified types are always used as the rightmost type of any derivation (the type given to the rightmost symbol in a derivation) and when the other types are never used as the rightmost type of any derivation (the types given to the other symbols). Technically, in the following theorem, the initial lexicon is transformed into an equivalent one for which we are sure that each type in the lexicon is always or never used as the rightmost type (but not both). A barrier is then added on the types that always appear on the rightmost type of any derivation.

Theorem 3. *Let G be a CDG$_b$. There exists an equivalent grammar (without empty head) $G' = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$ such that for every string $w_1 \cdots w_n \in W^*$ and every proof $\gamma_1 \cdots \gamma_n \vdash^*_b [S]^Q$ where $\gamma_1 \in \lambda(w_1), \dots \gamma_n \in \lambda(w_n)$, then the rightmost type $\gamma_n = B^{P\updownarrow}$, where B is a basic dependency type and P is a potential.*

The proof uses Theorem 2.

4.4. Context-free grammars and CDG extended with barriers

On the one hand, we consider *CF* as in Definition 10 in [1] applied to build a context-free grammar from a CDG$_b$. On the other hand, we can extend the definition of *CDG* in [1] (Definition 12, unchanged) to the case with barriers. We then get context-free lemmas with corollaries that are useful to show some AFL properties:

Corollary 1. *Let $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$ be a CDG$_b$ with $CF(G) = (\Sigma_1, N_1, S_1, \mathcal{P}_1)$, $S_1 = S$: $w_1 ... w_n \in L(G)$ iff $\exists P_1 ... \exists P_n : w_1^{P_1} ... w_n^{P_n} \in L(CF(G))$ and $P_1 ... P_n$ is balanced.*

Corollary 2. *Let $G_1 = (\Sigma_1, N_1, S_1, \mathcal{P}_1)$ be a cf-grammar in Greibach normal form, where the elements of Σ_1 are of the form w^P (where P in w^P is a CDG$_b$ potential) with $CDG(G_1) = (W', C', V', S', \lambda')$, $S' = S_1$:*

$w_1 ... w_n \in L(CDG(G_1))$ iff $\exists P_1 ... \exists P_n : w_1^{P_1} ... w_n^{P_n} \in L(G_1)$ and $P_1 ... P_n$ is balanced.

[4]Here, a barrier is added on the right of the potential of types of the lexicon but symmetricaly it is also possible to add it on the left of potential

4.5. Main category lemma

Lemma 1. *For every $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$ in CDG$_b$, there exists $G' = (W, \mathbf{C} \cup \{S'\}, \mathbf{V}, S', \lambda')$ which has "independent main category" and such as G and G' are equivalent (same languages).*

5. AFL CLOSURE PROPERTIES

We first group some properties that can be shown following the same approach as in [1].

Theorem 4.
[Union] *If $L_1 \in \mathcal{L}_k(CDG_b)$ and $L_2 \in \mathcal{L}_k(CDG_b)$, then $L_1 \cup L_2 \in \mathcal{L}_k(CDG_b)$.*
[ϵ-free homomorphisms] *If $L \in \mathcal{L}_k(CDG_b)$ is a language over W, and h is an ϵ-free homomorphism from W^+ to Σ^+, then $h(L) \in \mathcal{L}_k(CDG_b)$.*
[Inverses of homomorphisms] *If $L \in \mathcal{L}_k(CDG_b)$ and h is an homomorphism from Δ^* to W^*, then $h^{-1}(L) \in \mathcal{L}_k(CDG_b)$.*
[Intersection with regular sets] *If $L \in \mathcal{L}_k(CDG_b)$, and R is a regular language, then $L \cap R \in \mathcal{L}_k(CDG_b)$.*

5.1. Concatenation

The concatenation of languages defined by a CDG extended with barriers is also a CDG extended with barriers. For CDG (without barrier), [1] needs that the valency names of the grammars are disjoined. Thus, it isn't possible to have non-projective dependencies between the different parts in the concatenation. However, a consequence is that the valency complexity of the resulting grammar increases. In contrast, with CDG$_b$, there exists another construction that uses barriers to stop non-projective dependencies between the different parts of the concatenation. The construction needs a CDG$_b$ with a barrier on the rightmost types as it is explained in Theorem 3.

Theorem 5. *If $L_1 \in \mathcal{L}_k(CDG_b)$ and $L_2 \in \mathcal{L}_k(CDG_b)$, then $L_1 \cdot L_2 \in \mathcal{L}_k(CDG_b)$.*

Proof. See Annex A.2. □

5.2. Kleene plus

Kleene plus is an extension of the concatenation to an unlimited number of copies of the initial language. With CDG without barrier, it is not possible to restrict non-projective dependencies between the copies thus [1] wasn't able to propose a construction (the authors conjecture that Kleene plus isn't an internal operation in $\mathcal{L}(CDG)$). However, with barriers, it is possible to limit non-projective dependencies. Similarly with the construction for concatenation, we start with a CDG$_b$ with a barrier on the rightmost type as it is explained in Theorem 3.

Theorem 6. *If $L \in \mathcal{L}_k(CDG_b)$, then $L^+ \in \mathcal{L}_k(CDG_b)$.*

Proof. The proof is close to the proof for the concatenation of two languages. Let $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda) \in \mathcal{L}_k(CDG_b)$. Using Lemma 1 and Theorem 3 we may suppose that S cannot be used as an argument of a type and that there is a barrier on the right of the rightmost type in every proof ending with the axiom.
Let us define the grammar $G' = (W, \mathbf{C}, \mathbf{V}, S, \lambda \cup \lambda')$ where λ' is the lexicon λ where each type $[\alpha \backslash S / \beta]^P$ is replaced by the type $[\alpha \backslash S/S/\beta]^P$. Then, $G' \in \mathcal{L}_k(CDG_b)$ and $L(G') = L(G)^+$.
$[\Leftarrow]$ $L(G)^+ \subseteq L(G')$ In fact, for $n > 0$, we can transform n proofs of $\Gamma_1 \vdash_b^* [S]^{P_1}, \ldots, \Gamma_n \vdash_b^* [S]^{P_n}$ that generate n strings $x_1, \ldots, x_n$ in G ($P_1, \ldots, P_n$ are empty or contain only barriers) into

$n-1$ proofs of $\Gamma'_1 \vdash^*_b [\,S/S\,]^{P_1}, \ldots, \Gamma'_{n-1} \vdash^*_b [\,S/S\,]^{P_{n-1}}$ with types in λ'_1 (S is replaced by S/S) and an inchanged proof $\Gamma_n \vdash^*_b [\,S\,]^{P_n}$ with types in λ. These n proofs can be put together to define a proof of $\Gamma'_1 \cdots \Gamma'_{n-1}\Gamma_n \vdash^*_b [\,S\,]^{P_1 \cdots P_n}$ that generates the string $x_1 \cdots x_n$ in $L(G')$.
$[\Rightarrow]$ $L(G') \subseteq L(G)^+$ Let $x \in L(G')$. It exists $\Gamma \in (\lambda \cup \lambda')(x)$ such that $\Gamma \vdash_b [\,S\,]^P$ where $P = \updownarrow \cdots \updownarrow$ (P is empty or contains only barriers). Using Lemma 1, there exists $n > 0$ such that there are n types with head S in Γ. Because in the proof, all the S must be canceled except one, the first $n-1$ types must be $[\,\alpha_1 \backslash S/S/\beta_1\,]^{P_1}, \ldots, [\,\alpha_{n-1} \backslash S/S/\beta_{n-1}\,]^{P_{n-1}}$ (from λ') and the last one must be $[\,\alpha_n \backslash S/\beta_n\,]^{P_n}$ (from λ). Each S on the heads of the types are canceled by the preceding type. These cancelation steps can be postponed in the proof in order to be the last steps on basic dependency types and it is possible to postpone the steps on potential after the last cancelation of S: it exists a proof of $\Gamma \vdash^*_b [\,S/S\,]^{P_1} \cdots [\,S/S\,]^{P_{n-1}}[\,S\,]^{P_n} \vdash^*_b [\,S\,]^{P_1 \cdots P_n} \vdash^*_b [\,S\,]^P$. Now, Γ can be split in n parts such that $\Gamma_1 \vdash^*_b [\,S/S\,]^{P_1}, \ldots, \Gamma_{n-1} \vdash^*_b [\,S/S\,]^{P_{n-1}}$ and $\Gamma_n \vdash^*_b [\,S\,]^{P_n}$ that correspond to n parts $x_1, \ldots, x_n$ of x. When we replace S/S by S everywhere in the proofs of $\Gamma_1 \vdash^*_b [\,S/S\,]^{P_1}, \ldots, \Gamma_{n-1} \vdash^*_b [\,S/S\,]^{P_{n-1}}$, we obtain $n-1$ proofs of $\Gamma'_1 \vdash^*_b [\,S\,]^{P_1}, \ldots, \Gamma'_{n-1} \vdash^*_b [\,S\,]^{P_{n-1}}$ where $\Gamma'_1 \in \lambda(x_1), \ldots, \Gamma'_{n-1} \in \lambda(x_{n-1})$. Using Theorem 3, it means that $P_1 = P'_1\updownarrow, \ldots, P_{n-1} = P'_{n-1}\updownarrow$. Because $P_1 \cdots P_n = P'_1\updownarrow \cdots P'_{n-1}\updownarrow P_n$ is balanced, $P_1, \ldots, P_n$ must also be balanced. The proofs $\Gamma'_1 \vdash^*_b [\,S\,]^{P_1}, \ldots, \Gamma_{n-1} \vdash^*_b [\,S\,]^{P_{n-1}}$ and $\Gamma_n \vdash^*_b [\,S\,]^{P_n}$ can be completed with $\mathbf{D^l}$ and $\mathbf{D^r}$ steps such that we have proofs of $\Gamma'_1 \vdash^*_b [\,S\,]^{\updownarrow \cdots \updownarrow}, \ldots, \Gamma'_{n-1} \vdash^*_b [\,S\,]^{\updownarrow \cdots \updownarrow}$ and $\Gamma_n \vdash^*_b [\,S\,]^{\updownarrow \cdots \updownarrow}$: $x_1 \in L(G), \ldots, x_n \in L(G)$. $\square$

6. CONCLUSION AND OPEN QUESTIONS

In this paper we have considered the framework of categorial dependency grammars used in the field of natural language processing, with an interest in their formal properties. Whereas the closure problem under iteration is open for the original version of CDG, our approach is to propose an extension that fullfills the closure properties, without an increase in parsing complexity (for lack of space, our parsing algorithm for CDG$_b$ is not provided here). In that perspective, we have added a barrier mechanism, reflected essentially in types (attached to words) and rules that govern the parsing derivations. We have shown that the new class yields an Abstract Family of Languages, which is of interest for modular grammar constructs. Our AFL results also hold for each subclass $\mathcal{L}_k(CDG_b)$ where the number of valency names is bound by k. As compared to a former extension of CDG that yields an AFL, called multimodal (mmCDG), our proposal is closer to CDG, and avoids a complexity issue.

We also do not know how to characterize the expressive power of the extended version. We leave these open questions for future work.

REFERENCES

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

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

[3] I. Mel'čuk. *Dependency Syntax*. SUNY Press, Albany, NY, 1988.

[4] Denis Béchet and Annie Foret. Categorial dependency grammars: Analysis and learning. In Roussanka Loukanova, Peter LeFanu Lumsdaine, and Reinhard Muskens, editors, *Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, volume 1081 of *Studies*

in Computational Intelligence, pages 31–56. Springer, Cham, 2023. Edited results of LA-CompLing2021.

[5] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991.

[6] Makoto Kanazawa. Abstract families of abstract categorial languages. *Electron. Notes Theor. Comput. Sci.*, 165:65–80, 2006.

[7] Takashi Matsumura, Hiroyuki Seki, Mamoru Fujii, and Tadao Kasami. The generative power of multiple context-free grammars and head grammars. *Systems and Computers in Japan*, 22(4):41–56, 1991.

[8] Michael I. Dekhtyar, Alexander Ja. Dikovsky, and Boris Karlov. Iterated dependencies and kleene iteration. In Philippe de Groote and Mark-Jan Nederhof, editors, *Formal Grammar - 15th and 16th International Conferences, FG 2010, Copenhagen, Denmark, August 2010, FG 2011, Ljubljana, Slovenia, August 2011, Revised Selected Papers*, volume 7395 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 2010.

[9] Alexander Dikovsky. Multimodal categorial dependency grammars. In *Proc. of the 12th Conference on Formal Grammar*, pages 1–12, Dublin, Ireland, 2007.

ANNEX

A. DETAILS OF PROOF

A.1. Proof of Theorem 2

Proof. Let $G = (W, \mathbf{C}, \mathbf{V}, S, \lambda)$ be a CDG$_b$. Let us consider the grammar $G' = (W, \mathbf{C} \cup \mathbf{C}' \cup \{E\}, \mathbf{V}, S, \lambda')$ where $\mathbf{C}' = \{d' \ : \ d \in \mathbf{C}\}$ (we suppose that $\mathbf{C} \cap \mathbf{C}' = \emptyset$ and $E \notin \mathbf{C} \cup \mathbf{C}'$) and λ' is defined as follows:

- if $\lambda : w \mapsto [\, l_m \setminus \cdots \setminus l_1 \setminus \varepsilon \,/\, r_1 \,/\, \cdots \,/\, r_n \,]^P$ (empty head), then

$\lambda' : w \mapsto [\, E^* \backslash L_m \backslash E^* \backslash \cdots \backslash E^* \backslash L_1 \backslash E^* \backslash E / E^* / r_1 / E^* / \cdots / E^* / R_n / E^* \,]^P$

and for each $h' \in \mathbf{C}'$,

$\lambda' : w \mapsto [\, E^* \backslash L_m \backslash E^* \backslash \cdots E^* \backslash L_1 \backslash E^* \backslash h' / E^* / R_1 / E^* / \cdots / E^* / R_n / E^* \,]^P$

where for $i = 1, \ldots m$, $L_i = l_i$ if l_i isn't iterated or $L_i = d'^*$ if $l_i = d^*$ (similarly for R_j, $1 \leq j \leq n$).

- if $\lambda : w \mapsto [\, l_m \setminus \cdots \setminus l_1 \setminus h \,/\, r_1 \,/\, \cdots \,/\, r_n \,]^P$ (h not empty), then

$\lambda' : w \mapsto [\, E^* \backslash L_m \backslash E^* \backslash \cdots \backslash E^* \backslash L_1 \backslash E^* \backslash h / E^* / r_1 / E^* / \cdots / E^* / r_n / E^* \,]^P$

and

$\lambda : w \mapsto [\, E^* \backslash L_m \backslash E^* \backslash \cdots \backslash E^* \backslash L_1 \backslash E^* \backslash h' / E^* / r_1 / E^* / \cdots / E^* / r_n / E^* \,]^P$

where for $i = 1, \ldots m$, $L_i = l_i$ if l_i isn't iterated or $L_i = d'^*$ if $l_i = d^*$ (similarly for R_j, $1 \leq j \leq n$).

G' is a CDG$_b$ without empty head. A derivation using its types cannot use rules $\mathbf{L}^{\mathbf{l}}_{\varepsilon}, \mathbf{L}^{\mathbf{r}}_{\varepsilon}$: it is a derivation in $\vdash^*_H$.

Let us prove that $L(G) = L(G')$. A derivation ρ of $\Gamma \vdash^* [\, S \,]^P$ for the generation of a string $x \in L(G)$ can be transformed into a derivation ρ' of $\Gamma' \vdash^*_H [\, S \,]^P$ for the same string in $L(G')$ and conversely.

In ρ each step corresponding to rule $\mathbf{L}^{\mathbf{l}}_{\varepsilon}$ (or rule $\mathbf{L}^{\mathbf{r}}_{\varepsilon}$) is replaced by a step using rule $\mathbf{L}^{\mathbf{l}}$ (or rule $\mathbf{L}^{\mathbf{r}}$) when the step occurs outside the scope of an iterated type. If the step is inside cancelations of several types by the same iterated type d, the step is replaced by a step of rule $\mathbf{I}^{\mathbf{l}}$ (or rule $\mathbf{I}^{\mathbf{r}}$) on type d'. Each step corresponding to rule $\mathbf{I}^{\mathbf{l}}$ (or rule $\mathbf{I}^{\mathbf{r}}$) on type d is replaced by a step of rule $\mathbf{I}^{\mathbf{l}}$ (or rule $\mathbf{I}^{\mathbf{r}}$) on type d'. Steps corresponding to rules $\mathbf{\Omega}^{\mathbf{l}}$ and $\mathbf{\Omega}^{\mathbf{r}}$ are added in order to eliminate the E^* arguments.

In the other direction, if we start with a derivation ρ' of $\Gamma' \vdash^*_H [\, S \,]^P$ for the same string in $L(G')$, we apply a reverse transformation. In this case, the added steps that eliminate the E^* arguments simply disappear. □

A.2. Proof of Theorem 5

Proof. Let $G_1 = (W, \mathbf{C}_1, \mathbf{V}, S_1, \lambda_1) \in \mathcal{L}_k(CDG_b)$ and $G_2 = (W, \mathbf{C}_1, \mathbf{V}, S_2, \lambda_2) \in \mathcal{L}_k(CDG_b)$. We may suppose that the set of symbols W, and the set of valency names $\mathbf{V}$ are the same for both grammars. We also may suppose that $\mathbf{C}_1 \cap \mathbf{C}_2 = \emptyset$. Finally, using Lemma 1 and Theorem 3 we may suppose that S_1 and S_2 cannot be used as an argument of a (iterated or not) type and that for G_1, there is a barrier on the right of the rightmost type in every derivation of a string of W^* ending in $[\, S_1 \,]^P$.

Let us consider the grammar $G = (W, \mathbf{C}_1 \cup \mathbf{C}_2, \mathbf{V}, S_1, \lambda'_1 \cup \lambda_2)$ where the lexicon λ'_1 is the lexicon λ_1 where each type $[\, \alpha \setminus S_1 \,/\, \beta \,]^P$ is replaced by the type $[\, \alpha \setminus S_1 \,/\, S_2 \,/\, \beta \,]^P$. Then, $G \in \mathcal{L}_k(CDG_b)$ and $L(G) = L(G_1) \cdot L(G_2)$.

$[\Leftarrow]$ $L(G_1) \cdot L(G_2) \subseteq L(G)$

In fact, a proof $\Gamma_1 \vdash^*_b [\, S_1 \,]^{P_1}$ that generates a string x_1 in $L(G_1)$ can be transformed into a proof of $\Gamma'_1 \vdash^*_b [\, S_1 \,/\, S_2 \,]^{P_1}$ with types of λ'_1 (S_1 is replaced by $S_1 \,/\, S_2$). With a proof of $\Gamma_2 \vdash^*_b [\, S_2 \,]^{P_2}$ that generates a string x_2 in $L(G_2)$, we can define a proof of $\Gamma'_1\Gamma_2 \vdash_b^* [\, S_1 \,/\, S_2 \,]^{P_1}[\, S_2 \,]^{P_2} \vdash_b S_1^{P_1 P_2}$ that generates $x_1 x_2$ in G.

$[\Rightarrow]$ $L(G) \subseteq L(G_1) \cdot L(G_2)$

Let $x \in L(G)$. It exists $\Gamma \in (\lambda'_1 \cup \lambda_2)(x)$ such that $\Gamma \vdash_b^* [\, S_1 \,]^P$ where $P = \updownarrow \cdots \updownarrow$ (P is empty or contains only barriers). Using Lemma 1, in Γ, there is exactly one type with head S_1 (from λ'_1). The type is of the form $[\, \alpha \setminus S_1 \,/\, S_2 \,/\, \beta \,]^P$. Thus there is also exactly one type with head S_2 (from λ_2). These are the only occurences of S_1 and S_2 in Γ (as head or argument). The type of head S_2 must be on the right of the type $[\, \alpha \setminus S_1 \,/\, S_2 \,/\, \beta \,]^P$ in Γ. In the proof S_2 must be canceled by a subtype of this type. This step can be postponed in the proof to be the last step on basic dependency types: there exists two potentials $P1$ and P_2 such that $\Gamma \vdash_b^* [\, S_1 \,/\, S_2 \,]^{P_1}[\, S_2 \,]^{P_2} \vdash_b [\, S_1 \,]^{P_1 P_2} \vdash_b^* [\, S_1 \,]^P$. Moreover, it is also possible to postpone the steps on potential after the cancelation of S_2: We can suppose that the proof $\Gamma \vdash_b^* [\, S_1 \,/\, S_2 \,]^{P_1}[\, S_2 \,]^{P_2} \vdash_b [\, S_1 \,]^{P_1 P_2}$ doesn't use $\mathbf{D^l}$ or $\mathbf{D^r}$. Γ can be split in two parts such that $\Gamma_1 \vdash_b^* [\, S_1 \,/\, S_2 \,]^{P_1}$ and $\Gamma_2 \vdash_b^* [\, S_2 \,]^{P_2}$ that correspond to two parts x_1 and x_2 of x. We can prove that the types of Γ_1 must come from λ'_1 (because it ends with $[\, S_1 \,/\, S_2 \,]^{P_1}$) and the types of Γ_2 must come from λ_2 (because it ends with $[\, S_2 \,]^{P_2}$). Now, when we replace $S_1 \,/\, S_2$ by S_1 everywhere in the proof $\Gamma_1 \vdash_b^* [\, S_1 \,/\, S_2 \,]^{P_1}$, we obtain a proof $\Gamma'_1 \vdash_b^* [\, S_1 \,]^{P_1}$ where $\Gamma'_1 \in \lambda_1(x_1)$. Using Theorem 3, it means that $P_1 = P'_1 \updownarrow$. Because $P_1 P_2 = P'_1 \updownarrow P_2$ is balanced, P_1 and P_2 must also be balanced. The proofs $\Gamma'_1 \vdash_b^* [\, S_1 \,]^{P_1}$ and $\Gamma_2 \vdash_b^* [\, S_2 \,]^{P_2}$ can be completed with $\mathbf{D^l}$ and $\mathbf{D^r}$ steps such that we have proofs of $\Gamma'_1 \vdash_b^* [\, S_1 \,]^{\updownarrow \cdots \updownarrow}$ and $\Gamma_2 \vdash_b^* [\, S_2 \,]^{\updownarrow \cdots \updownarrow}$: $x_1 \in L(G_1)$ and $x_2 \in L(G_2)$. $\square$

AUTHORS

Denis Béchet obtained his PhD from the University of Paris 7. Currently, he is an Assistant Professor in the Department of Computer Sciences at Nantes University. His area of interests includes Natural Language Processing and Formal Grammars.

Annie Foret obtained her PhD from the University of Paris 7 and her HDR from the University of Rennes. Currently, she is an Assistant Professor in Computer Sciences at Rennes University. Her area of interests includes Natural Language Processing, Formal Grammars and Data Knowledge Management in the DKM Department at IRISA.