Iterated Dependencies and Kleene Iteration

Michael Dekhtyar[1], Alexander Dikovsky[2], and Boris Karlov[1] ⋆

[1] Dept. of Computer Science, Tver State University, Tver, Russia, 170000.
Michael.Dekhtyar@tversu.ru, bnkarlov@gmail.com
[2] LINA CNRS UMR 6241, Université de Nantes,
Alexandre.Dikovsky@univ-nantes.fr

Abstract. Categorial Dependency Grammars (CDG) is a class of simple and expressive categorial grammars defining projective and discontinuous dependency structures in a strongly compositional way. They are more expressive than CF-grammars, are polynomial time recognizable and different from the mildly context sensitive grammars. CDG languages are proved to be closed under all AFL operations, but `iteration`. In this paper, we explain the connection between the iteration closure and the iterated dependencies (optional repeatable dependencies, inevitable in dependency syntax) and show that the CDG extended by a natural multimodal rule define an AFL, but the membership problem in this extended family is NP-complete.
Keywords: Dependency Grammar, Categorial Dependency Grammar, Iterated Dependency, Iteration.

1 Introduction

In this paper are studied `iterated`, i.e. optional repeatable dependencies, such as `modifier` dependencies of nouns (e.g. *optional* $\overset{modif}{\longleftarrow}$ *dependencies* and `circumstantial` dependencies of verbs (e.g. $fits \overset{circ}{\longrightarrow} well$). The main question is whether the dependency grammars expressing such dependencies generate languages closed under Kleene iteration, or more generally, is there a direct connection between the former and the latter. It should be made clear what do we mean by "express iterated dependencies". In fact, in the traditional linguistics it is generally accepted that the ultimately repeatable modifiers / circumstantials share the same governor (e.g. see [12]). I.e., the adequate dependency structure for *a tall blond young girl* is that in Fig. 1(a) and not that in Fig. 1(b).[3]

⋆ This work was sponsored by the Russian Fundamental Studies Foundation (Grants No. 10-01-00532-a and 08-01-00241-a).

[3] By indirection, this means that the dependency structures corresponding to the traditional recursive types of modifiers / circumstantials used in the categorial grammars [1, 16], in Lambek grammar [11] and, more generally, in the type logical grammars interfacing the semantics of Montague [14] are not adequate from the traditional linguistic point of view. In [13] this structural defect is amended using types extended with modalities, but the resulting calculus is computationally untractable.

As it concerns the grammars not expressing discontinuous (crossing) dependencies and generating exactly the context-free languages, such as for instance the link grammars [15], the question is trivially answered in the positive sense. For the (rare) grammars expressing discontinuous dependencies, there is no gen-

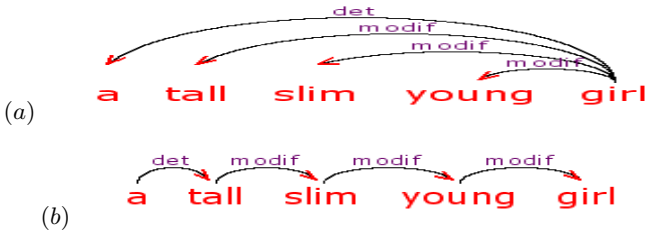

Fig. 1. Dependency structures: iterative vs. recursive

eral solution. E.g., for constraint grammars with NP-hard membership problem[4], such as topological dependency grammars [7], the answer is positive. But for polynomially analyzed dependency grammars expressing discontinuous dependencies the problem needs a specific solution for every particular class.

In this paper, we try to establish a connection between the iterated dependencies and the Kleene star closure in the class of categorial dependency grammars (CDG) [6]. CDG are categorial grammars based on a simple calculus of dependency types. They express unlimited discontinuous dependencies using simple polarized valencies' pairing rules and are polynomially parsed. CDG express iterated dependencies explicitly through iterated types of the form $t*$. For instance, in Fig. 1(a), for the word *girl* is used the type $[modif*\backslash det\backslash S]$, S being the axiom. It may seem that in the presence of such iterated types the generated languages are immediately closed under the Kleene iteration. This illusion has let down the authors of [3] who stated that the CDG languages constitute an AFL, and in particular are closed under iteration.[5] As we show below, in general, the direct closure construction doesn't work because of the CDG's valency pairing rule. We arrive to find rather a minimal modality extension of the basic CDG for which we finally prove the iteration closure property. However, the resulting extended CDG turn out to be NP-complete.

[4] Membership problem in a family $\mathcal{F}$ is the problem $w \in L(G)$ for a given grammar $G \in \mathcal{F}$ (not to confuse with the "uniform membership problem" $\{< w, G > | w \in L(G), G \in \mathcal{F}\}$).

[5] This is the only assertion in [3] stated without proof because of its "evidence".

2 Basics of Dependency Structures

Tesnière [17] was the first who systematically described the sentence structure in terms of named binary relations between words (`dependencies`). When two words w_1 and w_2 are related in a sentence through dependency relation d (denoted $w_1 \xrightarrow{d} w_2$), w_1 is the `governor` (also called `head`) and w_2 is the `subordinate`. Intuitively, the dependency d encodes constraints on lexical and grammatical features of w_1 and w_2, on their precedence order, pronominalization, context, etc. which together mean that "w_1 licenses w_2" (see [12] for a detailed exposition). A `dependency structure` (`DS`) is a graph of dependency relations between words in the sentence. For instance, the sentence *In the beginning was the Word* has the DS in Fig. 2, in which *was* $\xrightarrow{pred}$ *Word* stands for the predicative dependency between the copula *was* and the subject *Word*. There is no general agreement on the notion of DS: sometimes it is separated from the precedence order in the sentence, sometimes it is linearly ordered by the precedence, most people require it be a tree (the tradition going back to [17]), some others do not (cf. [8]) because without this constraint one can define mixed structures (e.g. taking in account the co-reference). When a DS is a tree it is called `dependency tree` (`DT`).

CDG use DS linearly ordered by the precedence order in the sentence. Without this order some fundamental properties of DS cannot be expressed. This is the case of one of the most important properties of DS, called `projectivity`. This property is expressed in terms of the `immediate dominance` relation: $w_1 \Rightarrow w_2 \equiv \exists d\ (w_1 \xrightarrow{d} w_2)$ and of its reflexive-transitive closure $\Rightarrow^*$ called `dominance`. A DT of a sentence x is `projective` if, for every word w in x, the set of all words dominated by w: $proj(w) = \{w' \parallel w \Rightarrow^* w'\}$ (called `projection` of w) is an `interval` of x with respect to the precedence order. In all languages, the majority of DS are projective DT (an example is given in Fig. 2). But even if this

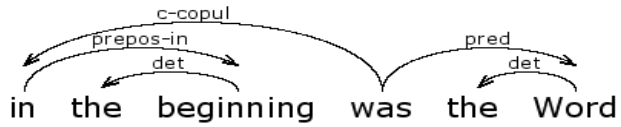

Fig. 2. A projective DT

is true, the projectivity is not a norm. Non-projective dependencies are often due to discontinuous constructions such as comparatives (cf. *more..than* in English or negation *ne..pas* in French). They are also caused by verb complements' dislocation, clefting, scrambling, conversion of complements into clitics and other regular constructions marking for a communicative structure of sentences. We

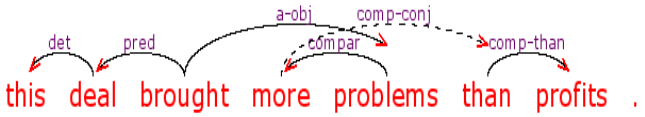

Fig. 3. A non-projective DT in English

show two examples of non-projective DT in Figs. 3, 4.

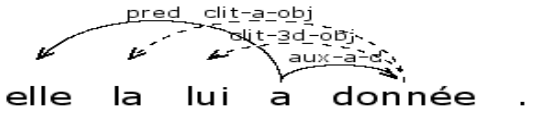

Fig. 4. A non-projective DT in French (**she* it_{FEM} *to-him has given*)

Definition 1. *Let* $w = a_1 \dots a_n$ *be a string,* W *be the set of all occurrences of symbols in* w *and* $C = \{d_1, \dots, d_m\}$ *be a set of* `dependency names`*. A graph* $D = (W, E)$ *with labeled arcs is a* `DS` *of* w *if it has a* `root`*, i.e. a node* $a_0 \in W$ *such that* (i) *for any node* $a \in W$, $a \neq a_0$*, there is a path from* a_0 *to* a *and* (ii) *there is no arc* (a', d, a_0).[6] *An arc* $(a_1, d, a_2) \in E$ *is called* `dependency` d *from* a_1 *to* a_2*. The linear order on* W *induced by* w *is the* `precedence order` *on* D*.*

3 Categorial Dependency Grammars

As all categorial grammars, the `categorial dependency grammars` (`CDG`) may be seen as assignments of dependency types to words. Every dependency type assigned to a word w defines its possible `local neighborhood` in grammatically correct DS. The neighborhood of w consists of the `incoming` dependency, i.e. the dependency relation d through which w is subordinate to a word G, its `governor`, and also of a sequence of `outgoing` dependencies, i.e. the dependency relations d_i through which w governs a subordinate word w_i. For instance, the type assignment:
$in \mapsto [c{-}copul/prepos{-}in]$, $the \mapsto [det]$, $beginning \mapsto [det\backslash prepos{-}in]$,
$was \mapsto [c{-}copul\backslash S/pred]$, $Word \mapsto [det\backslash pred]$
determines the DS in Fig. 2. In particular, the type $[c{-}copul/\ prepos{-}in]$ of in defines its local neighborhood, where $c{-}copul$ is the incoming dependency and

[6] Evidently, every DS is connected and has a unique root.

prepos−in is the right outgoing dependency. The verb *was* in the root has the head type S which serves as the grammar's `axiom`. CDG use iteration to express all kinds of repetitive dependencies and in particular the coordination relations. This provides more adequate DS than those traditionally defined in categorial grammars through recursive types $[X/X]$. For instance, the type assignment:
$a \mapsto [det]$, *tall*, *slim*, *young* $\mapsto [modif]$, $girl \mapsto [modif * \backslash det\ S]$ determines the DT in Fig. 1(a). Remark that in the canonical categorial grammars the types of articles and adjectives are functional.

In CDG, the non-projective dependencies are expressed using so called polarized valencies. Namely, in order that a word G may govern through a discontinous dependency d a word D situated somewhere on its right, D should have a type declaring the positive valency $\nearrow d$, whereas its subordinate D should have a type declaring the `negative` valency $\searrow d$. Together these `dual` valencies define the discontinous dependency d. By the way, the pairing itself of dual valencies is not enough to express the constraints of `adjacency` of the distant subordinate to a host word (`anchor constraints`). For this, in CDG are used the `anchor types` of the form $\#(\searrow d)$ treated in the same way as the local dependencies. So the general form of CDG dependency types is $[l_1 \backslash l_2 \backslash \ldots \backslash H / \ldots / r_2 / r_1]^P$, where the `head type` H defines the incoming dependency, l_i and r_i are respectively the left and the right outgoing dependencies or anchors and P is the `potential`, i.e. a string of polarized valencies defining incoming and outgoing discontinuous long distance dependencies.[7] We show two examples of non-projective DT in Figs. 3, 4. E.g., the non projective DT in Fig. 4 is defined by the assignment:
elle $\mapsto [pred]$
la $\mapsto [\#(\swarrow clit{-}a{-}obj)]^{\swarrow clit\text{–}a\text{–}obj}$
lui $\mapsto [\#(\swarrow clit{-}3d{-}obj)]^{\swarrow clit\text{–}3d\text{–}obj}$
$a \mapsto [\#(\swarrow clit{-}3d{-}obj) \backslash \#(\swarrow clit{-}a{-}obj) \backslash pred \backslash S / aux]$
donnée $\mapsto [aux]^{\nwarrow clit\text{–}3d\text{–}obj \nwarrow clit\text{–}a\text{–}obj}$.

Definition 2. *Let* $\mathbf{C}$ *be a set of* `local dependency names` *and* $\mathbf{V}$ *be a set of* `valency names`.

The expressions of the form $\swarrow v$, $\nwarrow v$, $\searrow v$, $\nearrow v$, *where* $v \in \mathbf{V}$, *are called* `polarized valencies`. $\nwarrow v$ *and* $\nearrow v$ *are* `positive`, $\swarrow v$ *and* $\searrow v$ *are* `negative`; $\nwarrow v$ *and* $\swarrow v$ *are* `left`, $\nearrow v$ *and* $\searrow v$ *are* `right`. *Two polarized valencies with the same valency name and orientation, but with the opposite signs are* `dual`.

An expression of one of the forms $\#(\swarrow v)$, $\#(\searrow v)$, $v \in \mathbf{V}$, *is called* `anchor type` *or just* `anchor`. *An expression of the form* d^* *where* $d \in \mathbf{C}$, *is called* `iterated dependency type`.

Local dependency names, iterated dependency types and anchor types are `primitive types`.

An expression of the form $t = [l_m \backslash \ldots \backslash l_1 \backslash H / \ldots / r_1 \ldots / r_n]$ *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 or an anchor type, or empty, is called* `basic dependency type`. $l_1, \ldots, l_m$

[7] All subtypes being dependency names or iterated, this means that the CDG types are first order.

and $r_1, \ldots, r_n$ *are respectively left and right* `argument subtypes` *of* t*. If nonempty,* H *is called* `head subtype` *of* t *(or* `head type` *for short).*

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

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

CDG are defined using the following calculus of dependency types [8]

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

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

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

$\mathbf{D^l}$. $\alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1PP_2}$, if the potential $(\swarrow C)P(\nwarrow C)$ satisfies the following pairing rule **FA** (`first available`):

$$\mathbf{FA}: \quad P \text{ has no occurrences of } \swarrow C, \nwarrow C.$$

$\mathbf{L^l}$ is the classical elimination rule. Eliminating the argument subtype $C \neq \#(\alpha)$ it constructs the (`projective`) dependency C and concatenates the potentials. $C = \#(\alpha)$ creates the `anchor dependency`. $\mathbf{I^l}$ derives $k > 0$ instances of C. $\mathbf{\Omega^l}$ serves for the case $k = 0$. $\mathbf{D^l}$ creates `discontinuous dependencies`. It pairs and eliminates dual valencies with name C satisfying the rule **FA** to create the discontinuous dependency C.

Definition 3. *A* categorial dependency grammar (CDG)[9] *is a system* $G = (W, \mathbf{C}, S, \lambda)$*, 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), and* λ*, called* lexicon*, is a finite substitution on* W *such that* $\lambda(a) \subset \mathbf{CAT}(\mathbf{C}, \mathbf{V})$ *for each word* $a \in W$*.*

For a DS D *and a string* $x \in W^*$*, let* $G(D, x)$ *denote the relation:* D *is constructed in a proof* $\Gamma \vdash S$ *for some* $\Gamma \in \lambda(x)$*. Then the* language *generated by* G *is the set* $L(G) =_{df} \{w \parallel \exists D\ G(D, w)\}$ *and the* DS-language *generated by* G *is the set* $\Delta(G) =_{df} \{D \parallel \exists w\ G(D, w)\}$*.* $\mathcal{D}(CDG)$ *and* $\mathcal{L}(CDG)$ *will denote the families of DS-languages and languages generated by these grammars.*

Below we cite from [4, 3] several examples and facts showing that CDG are very expressive. Evidently, they generate all context-free languages. They can also generate non-CF languages.

Example 1. The CDG G_{abc} : $a \mapsto A^{\swarrow A}$, $[A\backslash A]^{\swarrow A}$, $b \mapsto [B/C]^{\nwarrow A}$, $[A\backslash S/C]^{\nwarrow A}$, $c \mapsto C$, $[B\backslash C]$ generates the language $\{a^n b^n c^n \mid n > 0\}$. E.g., $G_{abc}(D^{(3)}, a^3b^3c^3)$ holds for the DS in Fig 5 and the string $a^3b^3c^3$ due to the proof in Fig. 6.

Seemingly, $\mathcal{L}(CDG)$ is different from mildly context-sensitive languages [9, 18] generated by multi-component TAG, linear CF rewrite systems and some other grammars. $\mathcal{L}(CDG)$ contains non-TAG languages, e.g. $L^{(m)} = \{a_1^n a_2^n ... a_m^n \parallel n \geq$

[8] We show left-oriented rules. The right-oriented are symmetrical.

[9] They are called `generalized CDG` in [3] in order to distinguish them from CDG generating DT, which we do not consider here.

Fig. 5. DS for $a^3b^3c^3$

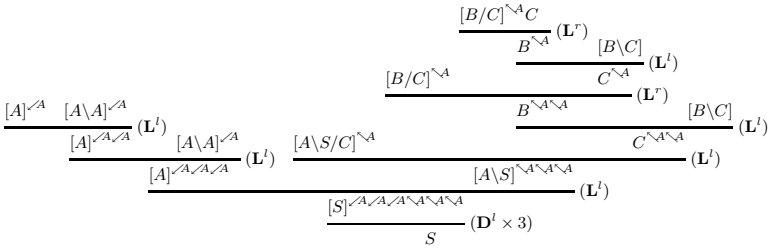

Fig. 6. DS correctness proof

1} for all $m > 0$. In particular, it contains the language $MIX = \{w \in \{a, b, c\}^+ \mid |w|_a = |w|_b = |w|_c\}$ [2], for which E. Bach has conjecture that it is not mildly CS. On the other hand, in [3] it is conjectured that this family does not contain the copy language $L_{copy} = \{xx \mid x \in \{a, b\}^*\}$, which is TAG. This comparison shows a specific nature of the valencies' pairing rule **FA**. This rule implies an important property of independence of basic types and of polarized valencies expressed in terms of `projections` of types and `''well-bracketing''` criteria for potentials.

For every type α and every sequence of types γ the `local projection` $\|\gamma\|_l$ and the `valency projection` $\|\gamma\|_v$ are defined as follows:
1. $\|\varepsilon\|_l = \|\varepsilon\|_v = \varepsilon$; $\|\alpha\gamma\|_l = \|\alpha\|_l\|\gamma\|_l$ and $\|\alpha\gamma\|_v = \|\alpha\|_v\|\gamma\|_v$.
2. $\|C^P\|_l = C$ et $\|C^P\|_v = P$ for every type C^P.

To speak about "well-bracketing" of potentials, it is useful to interpret $\swarrow d$ and $\nearrow d$ as `left brackets` and $\nwarrow d$ and $\searrow d$ as `right brackets`. Then a potential is `balanced` if it is well bracketed in the usual sense.

Let $\mathbf{c}$ be the projective core of the dependency calculus, consisting of the rules $\mathbf{L}$, $\mathbf{I}$ and $\mathbf{\Omega}$ and $\vdash_{\mathbf{c}}$ denote the provability relation in this sub-calculus. Then the `projections independence` property of CDG [3] is formulated as follows.

Theorem 1. *[3] For a CDG G with lexicon λ and a string x, $x \in L(G)$ iff there is $\Gamma \in \lambda(x)$ such that $\|\Gamma\|_l \vdash_{\mathbf{c}}^* S$ and $\|\Gamma\|_v$ is balanced.*

On this property resides a polynomial time parsing algorithm for CDG [3].

4 Problem of Iteration and a Multimodal Solution

As we saw, the types of CDG admit iterated subtypes. So it may seem that the family of CDG languages is trivially closed under Kleene iteration. To see why the straightforward construction does not work, let us consider the CDG:
$a \mapsto A^{\swarrow A},\ [A\backslash A]^{\swarrow A},$
$b \mapsto [B/C]^{\nwarrow A},\ [A\backslash S1/C]^{\nwarrow A},\ [A\backslash S/S1/C]^{\nwarrow A},$
$c \mapsto C,\ [B\backslash C]$
It may seem that $L(G) = L(G_{abc})L(G_{abc})$, but it is not so because it contains, for example, the string *aaabbccabbcc* which has the DS in Fig. 7. We see that this

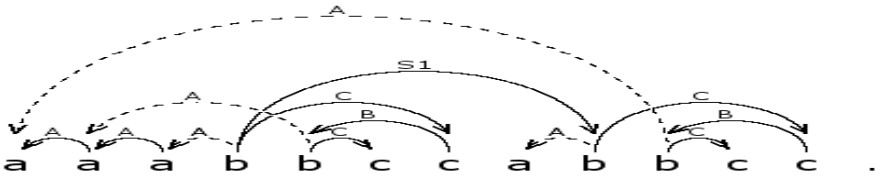

Fig. 7. DS of *aaabbccabbcc*

effect is due to the fact that dual valencies may sometimes be paired across the limits of concatenated / iterated strings and not within the limits, as needed. Of course, one can easily avoid this effect by renaming the valencies $\swarrow A$ and $\nwarrow A$. Indeed, this may work for concatenation and for any finite power $L(G_{abc})^k$, but this won't work for $L(G_{abc})^*$.

Now that the source of the problem is found, we will try to use possibly economical means to express the constraint of a "limit impenetrable for discontinuous dependencies". For that, we will follow the proposal of [5] where are introduced the so called multimodal CDG in which it is possible that every polarized valency has its own pairing rule (`pairing mode`).

Definition 4. *$G = (W, \mathbf{C}, S, \lambda, \mu)$ is a* multimodal CDG *(mmCDG) if $(W, \mathbf{C}, S, \lambda)$ is a CDG, in which are admitted empty head types ε, and μ is a function assigning to each polarized valency α a pairing principle $\mathbf{M}_\alpha$. There are rules $\mathbf{D}^l_{\mathbf{M}_\alpha}$ and $\mathbf{D}^r_{\mathbf{M}_\alpha}$ in the multimodal dependency calculus $\vdash_\mu$ for every valency α used in μ. The language (DS-language) generated by G using a set of modes M is denoted $L^M(G)$ ($\Delta^M(G)$). $mmCDG^M$ is the family of all such mmCDG.*

For instance, in [5] the calculus rule $\mathbf{D}^l$ is replaced by a new rule $\mathbf{D}_{\mathbf{FC}^l}$ in which in the place of the pairing rule **FA** is used the following pairing rule $\mathbf{FC}^l$ (`first cross`):

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

if $P_1(\swarrow C)P(\nwarrow C)$ satisfies the pairing rule

$\mathbf{FC}^l$: P_1 has no occurrences of $\swarrow C$ and P has no occurrences of $\nwarrow C$.

This rule was used to show that the so called `unlimited cross-serial dependencies` in Dutch are naturally expressed in mmCDG.

In this section we will show how the iteration problem can be resolved using multimodal CDG. For this, we will use `negative mode` pairing rules $\mathbf{FA}_{C:\pi(C)}$ which pair dual valencies C under the negative condition that the resulting discontinous dependency C `does not cross` the discontinuous dependencies belonging to a fixed list $\pi(C)$. More precisely, in the discontinuous dependency rule

$$\mathbf{D}_{\mathbf{FA}^l_{C:\pi(C)}} \qquad \alpha^{P_1(\swarrow C)P(\nwarrow C)P_2} \vdash \alpha^{P_1PP_2},$$

$(\swarrow C)P(\nwarrow C)$ satisfies the pairing rule $\mathbf{FA}_{C:\pi(C)}$:

P has no occurrences of $\swarrow C, \nwarrow C$ and also of $\swarrow A, \nwarrow A, \nearrow A, \searrow A$ for all $A \in \pi(C)$.

The mmCDG with this pairing rule will be denoted $(W, \mathbf{C}, S, \lambda, \mu, \pi)$.

Our purpose is to prove that the family $\mathcal{L}(mmCDG^{-FA})$ of languages generated by mmCDG with the negative mode pairing rules is closed under iteration.

First of all, we remark that the projections' independence property in Theorem 1 also holds for the mmCDG with the negative modes. Indeed, it is not difficult to see that the proof of this Theorem in [5] may be extended to mmCDG with the rules $\mathbf{FA}_{C:\pi(C)}$ in a straightforward way.

Then, as it shows the following Lemma, one can consider without loss of generality only $mmCDG$ in Greibach normal form.

Lemma 1. *[2, 10] For every CDG G there is an equivalent CDG* [10] *G' such that every type has one of the forms: $[A]^P$, $[A/B]^P$ or $[A/B/C]^P$ where B, C are primitive types different from S.*

Theorem 2. *$\mathcal{L}(mmCDG^{-FA})$ is closed under iteration.*

Proof. Let us suppose that $G = (W, \mathbf{C}, S, \lambda, \mu, \pi)$ is an mmCDG in the normal form. We will define from G a sequence of mmCDG $G_i = (W, \mathbf{C}_i, S, \lambda_i, \mu, \pi)$ by the following induction on i.
I. $i = 1$. $\lambda_1 = \lambda \cup \{w \mapsto [S/A'/\alpha]^P \mid (w \mapsto [S/A/\alpha]^P) \in \lambda\}$ for a new local dependency name $A' \in \mathbf{C}_1$.
II. $i > 1$. Let $A'_1, \ldots, A'_k \in \mathbf{C}_i$ be all new local dependency names in the grammar G_i. Let us first consider the auxiliary extended lexicon
$\lambda''_{i+1} = \lambda_i \ \cup \ \{w \mapsto [A'_j/B]^P \mid (w \mapsto [A_j/B]^P) \in \lambda_i, 1 \le j \le k\} \cup$
$\{w \mapsto [A'_j/B/C]^P \mid (w \mapsto [A_j/B/C]^P) \in \ \lambda_i, 1 \le j \le k\}$. Then let us set

$$\lambda_{i+1} = \lambda''_{i+1} \cup \{w \mapsto [A'_j/A'/\alpha]^P \mid (w \mapsto [A'_j/A/\alpha]^P) \in \lambda''_{i+1}\}.$$

New types A'_j, A' are added to $\mathbf{C}_{i+1}$. This construction converges to a mmCDG $G_m = (W, \mathbf{C}_m, S, \lambda_m, \mu, \pi)$, $m \le |\mathbf{C}|$. Let $b \notin \mathbf{C}_m$ be a new local dependency name. Let us consider an auxiliary mmCDG $G'_m = (W, \mathbf{C}_m \cup \{b\}, S, \lambda'_m, \mu', \pi')$ constructed from G_m as follows. In λ_m, every type $[A']^P$ is replaced by $[A']^{P\searrow b}$,

[10] Here `equivalent` corresponds to `weakly equivalent`, i.e. generating the same language (possibly not the same DS-language).

every type $[S/A'/\alpha]^P$ is replaced by $[S/A'/\alpha]^{\nearrow bP}$ and every type $[S]^P$ is replaced by $[S]^{\nearrow bP\searrow b}$. Let also $\pi'(A) = \pi(A) \cup \{b\}$ for all $A \in \mathbf{C}_m \cup \{b\}$. Now, the mmCDG G' defining the iteration of $L(G)$ can be defined as follows. $G' = (W, \mathbf{C}_m \cup \{b, S_0\}, S_0, \lambda', \mu', \pi')$, where $\lambda' = \lambda'_m \cup \{w \mapsto [S_0/S*/\alpha]^P \mid (w \mapsto [S/\alpha]^P) \in \lambda'_m\}$.

Let us prove that $L(G') = L(G)^*$.

Lemma 2. *If in G'_m $\gamma_1 \dots \gamma_n \vdash^* [A]^P$, where $A \in \mathbf{C}$, then there are no types A' in $\gamma_1 \dots \gamma_n$.*

Proof. By straightforward induction on n. E.g., if $\gamma_1 = [A/B]^{P_1}$, then $\gamma_2 \dots \gamma_n \vdash^* [B]^{P_1}$, $B \in \mathbf{C}$ and therefore, there are no types A' in $\gamma_2 \dots \gamma_n$.

Lemma 3. *Let $\gamma_1 \dots \gamma_n \vdash^* [A']^P$ in G'_m. Then $\gamma_n = [X']^{P'\searrow b}$ and there are no occurrences of b in the potentials of types $\gamma_1 \dots \gamma_{n-1}$.*

Proof. By induction on n.
When $n = 1$, $\gamma_1 = [X']^{P'\searrow b}$ and $P = P' \searrow b$ by construction.
Let $\gamma_1 \dots \gamma_{n+1} \vdash^* [A']^P$ in G'_m.
1) If $\gamma_1 = [A'/B']^{P_1}$, then $\gamma_2 \dots \gamma_{n+1} \vdash^* [B']^{P_2}$. By hypothesis, $\gamma_{n+1} = [X']^{P'\searrow b}$ and potentials of $\gamma_2 \dots \gamma_{n+1}$ do not contain b. P_1 does contain b by construction.
2) If $\gamma_1 = [A'/B'/C]^{P_1}$, then $\gamma_2 \dots \gamma_r \vdash^* [C]^{P_2}$ and $\gamma_{r+1} \dots \gamma_{n+1} \vdash^* [B']^{P_3}$ for some r. By hypothesis, $\gamma_{n+1} = [X']^{P'\searrow b}$ and potentials of $\gamma_{r+1} \dots \gamma_n$ do not contain b. By Lemma 2, $\gamma_2 \dots \gamma_r$ do not contain subtypes Y' so their potentials do not contain b by construction. P_1 also does not contain b by construction.
3) Other cases are impossible when the derived type has the form $[A']^P$.

Lemma 4. *Let $\Gamma = \gamma_1 \dots \gamma_n \in \lambda'_m(w)$ and $\Gamma \vdash^*_{G'_m} [A]^P$ or $\Gamma \vdash^*_{G'_m} [A']^{P\searrow b}$ for some $A \neq S$. Then there is $\Gamma' = \gamma'_1 \dots \gamma'_n \in \lambda(w)$ such that $\Gamma' \vdash^*_G [A]^P$.*

Proof. If $\Gamma \vdash^*_{G'_m} [A]^P$, then by Lemma 2 the types $\gamma_1, \dots, \gamma_n$ may also be assigned in G. So $\Gamma' = \Gamma$ in this case.
When $\Gamma \vdash^*_{G'_m} [A']^{P\searrow b}$, the proof proceeds by induction on n.
For $n = 1$, $\gamma_1 = [A']^{P\searrow b}$ and $\gamma'_1 = [A]^P$.
Let $\Gamma = \gamma_1 \dots \gamma_{n+1} \vdash^*_{G'_m} [A']^{P\searrow b}$ for some $A \neq S$.
1) $\gamma_1 = [A'/B']^{P_1}$. Then $\gamma_2 \dots \gamma_{n+1} \vdash^*_{G'_m} [B']^{P_2\searrow b}$. By hypothesis, there is a proof $\gamma'_2 \dots \gamma'_{n+1} \vdash^*_G [B]^{P_2}$. Let us set $\gamma'_1 = [A/B]^{P_1}$. Then $\gamma'_1 \dots \gamma'_{n+1} \vdash^*_G [A/B]^{P_1}[B]^{P_2} \vdash_G [A]^{P_1P_2} = [A]^P$.
2) Case $\gamma_1 = [A'/B'/C]^{P_1}$ is similar.

Lemma 5. *Let $\Gamma = \gamma_1 \dots \gamma_n \in \lambda(w)$, $\Gamma \vdash^*_G [A]^P$ and $A' \in \mathbf{C}'_m$. Then there is $\Gamma' = \gamma'_1 \dots \gamma'_n \in \lambda'_m(w)$ such that $\Gamma' \vdash^*_{G'_m} [A']^{P\searrow b}$.*

Proof. Induction on n.
For $n = 1$, $\gamma_1 = [A]^P$. So we can set $\gamma'_1 = [A']^{P\searrow b}$.
Let $\Gamma = \gamma_1 \dots \gamma_{n+1}$.
1) $\gamma_1 = [A/B]^{P_1}$. Then $\gamma_2 \dots \gamma_{n+1} \vdash^*_G [B]^{P_2}$, where $P_1P_2 = P$. As $A' \in \mathbf{C}'_m$, B'

is also added by construction: $B' \in \mathbf{C}'_m$. By hypothesis, there is $\gamma'_2 \dots \gamma'_{n+1}$ such that $\gamma_2 \dots \gamma_{n+1} \vdash^*_{G'_m} [B']^{P_2 \searrow b}$. Let us set $\gamma'_1 = [A'/B']^{P_1}$. Then $\gamma'_1 \dots \gamma'_{n+1} \vdash^*_{G'_m} [A'/B']^{P_1}[B']^{P_2 \searrow b} \vdash_{G'_m} [A']^{P_1 P_2 \searrow b} = [A]^{P \searrow b}$.
2) $\gamma_1 = [A/B/C]^{P_1}$. In this case, $\gamma_2 \dots \gamma_r \vdash^*_G [C]^{P_2}$, $\gamma_{r+1} \dots \gamma_{n+1} \vdash^*_G [B]^{P_3}$ and $P_1 P_2 P_3 = P$. All types of G not containing S remain in G'_m. Besides this, $B' \in \mathbf{C}'_m$. Therefore, there is $\gamma'_2 \dots \gamma'_r$ such that $\gamma'_2 \dots \gamma'_r \vdash^*_{G'_m} [C]^{P_2}$ $(\gamma'_i = \gamma_i)$ and $\gamma'_{r+1} \dots \gamma'_{n+1}$ such that $\gamma'_{r+1} \dots \gamma'_{n+1} \vdash^*_{G'_m} [B']^{P_3 \searrow b}$ (induction hypothesis). So $\gamma'_1 \dots \gamma'_{n+1} \vdash^*_{G'_m} [A'/B'/C]^{P_1}[C]^{P_2}[B']^{P_3 \searrow b} \vdash^*_{G'_m} [A']^{P_1 P_2 P_3 \searrow b} = [A']^{P \searrow b}$.

Lemma 6. $L(G'_m) = L(G)$.

Proof. [⇒]. Let $w = w_1 \dots w_n \in L(G'_m)$. Then there is $\gamma_1 \dots \gamma_n \in \lambda'_m(w)$ such that $\gamma_1 \dots \gamma_n \vdash^*_{G'_m} [S]$.
1) If $\gamma_1 = [S]^{\nearrow bP \searrow b}$, then we should just replace it by $[S]^P$.
2) Let $\gamma_1 = [S/A']^{\nearrow bP_1}$. Then $\gamma_2 \dots \gamma_n \vdash^*_{G'_m} [A']^{P_2 \searrow b}$ and the potential $P_1 P_2$ is balanced. By Lemma 4, there is $\gamma'_2 \dots \gamma'_n \in \lambda(w_2 \dots w_n)$ such that $\gamma'_2 \dots \gamma'_n \vdash^*_G [A]^{P_2}$. Let us set $\gamma'_1 = [S/A]^{P_1} \in \lambda(w_1)$. Then $\gamma'_1 \dots \gamma'_n \vdash^*_G [S/A]^{P_1}[A]^{P_2} \vdash_G [S]^{P_1 P_2} \vdash^*_G [S]$.
3) The case $\gamma_1 = [S/A'/B]^{\nearrow bP_1}$ is similar. So $w \in L(G)$.
[⇐]. Let $w = w_1 \dots w_n \in L(G)$. Then there is $\gamma_1 \dots \gamma_n \in \lambda(w)$ such that $\gamma_1 \dots \gamma_n \vdash^*_G [S]$.
1) If $\gamma_1 = [S]^P$, we can just replace it by $[S]^{\nearrow bP \searrow b}$.
Let us prove the case 3) $\gamma_1 = [S/A/B]^{P_1}$ (case 2) is similar). In this case, $\gamma_2 \dots \gamma_r \vdash^*_G [B]^{P_2}$ and $\gamma_{r+1} \dots \gamma_n \vdash^*_G [A]^{P_3}$ for some r and the potential $P_1 P_2 P_3$ is balanced. In G'_m w_1 has the type $\gamma'_1 = [S/A'/B]^{\nearrow bP_1} \in \lambda'_m(w_1)$. As all types without S are kept in G'_m, we have $\gamma_2 \dots \gamma_r \in \lambda'_m(w_2 \dots w_r)$ and $\gamma_2 \dots \gamma_r \vdash^*_{G'_m} [B]^{P_2}$. So we set $\gamma'_2 = \gamma_2, \dots, \gamma'_r = \gamma_r$. Besides this, by Lemma 5, there is $\gamma'_{r+1} \dots \gamma'_n \in \lambda'_m(w_{r+1} \dots w_n)$ such that $\gamma'_{r+1} \dots \gamma'_n \vdash^*_{G'_m} [A']^{P_3 \searrow b}$. Then $\gamma'_1 \dots \gamma'_n \vdash^*_{G'_m} [S/A'/B]^{\nearrow bP_1}[B]^{P_2}[A']^{P_3 \searrow b} \vdash^*_{G'_m} [S]^{\nearrow bP_1 P_2 P_3 \searrow b} \vdash^*_{G'_m} [S]$. Therefore, $w \in L(G'_m)$.

By this Lemma, it is now sufficient to prove that $L(G'_m)^* = L(G')$.
[⇒] $L(G'_m)^* \subseteq L(G')$. This inclusion is rather evident. If $x = x_1 \dots x_n \in L(G'_m)^*$ and $x_1, \dots, x_n \in L(G'_m)$, then there are type assignments $\Gamma_i \in \lambda'_m(x_i)$ such that $\Gamma_i \vdash^*_{G'_m} [S]$, $1 \le i \le n$. The first type in Γ_1 has the form $[S/\alpha]^{\nearrow bP_1}$. We will replace this type by $[S_0/S*/\alpha]^{\nearrow bP_1}$ obtaining a new sequence $\Gamma'_1 \in \lambda'(x_1)$ such that $\Gamma'_1 \vdash_{G'} [S_0/S*]$. Now we can assign to x the sequence $\Gamma'_1\Gamma_2 \dots \Gamma_n \in \lambda'(x)$ such that $\Gamma'_1\Gamma_2 \dots \Gamma_n \vdash^*_{G'} [S_0/S*][S] \dots [S] \vdash^*_{G'} [S_0]$.
[⇐] $L(G') \subseteq L(G'_m)^*$. $x \in L(G')$ means that there is $\Gamma \in \lambda'(x)$ such that $\Gamma \vdash^*_{G'} [S_0]$. We can decompose this proof into the subproofs which eliminate consecutive iterated S: $\Gamma = \Gamma_1 \dots \Gamma_n$, where $\Gamma_i \in \lambda'(x_i)$ $1 \le i \le n$, and $\Gamma_1 \vdash^*_{G'} [S_0/S*]^{P_1}$, $\Gamma_j \vdash^*_{G'} [S]^{P_j}$, $2 \le j \le n$. For all $i, 1 \le i \le n$, there also exist proofs $\Gamma_i \vdash^*_{G'_m} [S]^{P_i}$ in which the potentials P_i are balanced. Indeed, if $|x_i| > 1$, then $\Gamma_i = [S/\alpha_1]^{\nearrow bP_i^1}[\alpha_2]^{P_i^2} \dots [\alpha_{k-i}]^{P_i^{k-1}}[A']^{P_i^k \searrow b}$. The last type has the form $[A']^{P_i^k \searrow b}$ due to Lemma 3. By Lemmas 2,3 the potential $P_i^1 \dots P_i^k$ does not

contain occurrences of b. So by definition of the rule $\mathbf{FA}_{C:\pi(C)}$ this potential is balanced. Therefore, P_i is balanced too. If otherwise $|x_i| = 1$, then $P_i = \nearrow bP_i' \searrow b$ is also balanced. As a result, $\Gamma_i \vdash^*_{G_m'} [S]$, i.e. $x_i \in L(G_m')$ for all $i, 1 \leq i \leq n$, and so $x \in L(G_m')^*$.

Corollary 1. *The family of $mmCDG^{-FA}$-languages is an AFL.*

Proof. By Theorem 2 and Theorem 4 in [3].

5 Expressiveness of mmCDG with Negative Modes

The negative constraints used for the iteration closure turn out to be rather expressive. E.g., using such constraints one can generate an exponential length strings language.

Let $L_{exp} = \{1010^2 1 \dots 10^{2^n} 1 \mid n > 1\}$ and G_{exp} be the following grammar:
$1 \mapsto [S/A_1]^{\nearrow X},\ [D_1/C]^{\nearrow Y},\ [B/A]^{\searrow X \nearrow X},\ [D/C]^{\searrow Y \nearrow Y},\ [B/A_2]^{\searrow X \nearrow X},$
$[D/C_2]^{\searrow Y \nearrow Y},\ [A_2]^{\searrow X \searrow Y},\ [C_2]^{\searrow X \searrow Y}$
$0 \mapsto [A_1/D_1]^{\nearrow B \nearrow B},\ [A/A]^{\searrow A \nearrow B \nearrow B},\ [A/D]^{\searrow A \nearrow B \nearrow B},\ [C/C]^{\searrow B \nearrow A \nearrow A},$
$[C/B]^{\searrow B \nearrow A \nearrow A},\ [A_2/A_2]^{\searrow A},\ [C_2/C_2]^{\searrow B}$

In this mmCDG is used the **FA** pairing rule with two negative modalities: $\pi(X) = \{A\}, \pi(Y) = \{B\}$. The intuitive idea is that every 0 takes one negative valency ($\searrow A$ or $\searrow B$) and puts out two positive valencies ($\nearrow B$ or $\nearrow A$). A and B are alternated in order that zeros in the same block couldn't be linked by a dependency. In order that a zero were linked with another zero in the next block, the consecutive symbols 1 are linked by discontinuous dependencies X (for even blocks) or Y (for odd blocks). Due to the negative modalities π, X does not let pass dependency A and Y does not let pass B. As a result, $L(G_{exp}) = L_{exp}$. A formal proof of this equality is a consequence of the following fact.

Lemma 7. *Let $w \in L(G_{exp})$, $w = w'w''$ and $w' = 10^{i_1}1 \dots 10^{i_k}1$, where $i_j \geq 0$. Let $\Gamma \in \lambda_{G_{exp}}(w)$, $\Gamma \vdash^*_{G_{exp}} [S]$ and $\Gamma = \Gamma'\Gamma''$, where $\Gamma' = \lambda_{G_{exp}}(w')$ and $\Gamma'' = \lambda_{G_{exp}}(w'')$. Then:*
1) $i_j = 2^{j-1}$ for $1 \leq j \leq k$.
2) If $w'' \neq \varepsilon$ and k is odd, then $\|\Gamma'\|_v \vdash^* \nearrow X(\nearrow B)^{2^k} \nearrow Y$ [11] and to the last 1 in w' is assigned in Γ' one of types: $[D_1/C]^{\nearrow Y}$, $[D/C]^{\searrow Y \nearrow Y}$ or $[D/C_2]^{\searrow Y \nearrow Y}$.
3) If $w'' \neq \varepsilon$ and k is even, then $\|\Gamma'\|_v \vdash^* \nearrow Y(\nearrow A)^{2^k} \nearrow X$ and to the last 1 in w' is assigned in Γ' one of types: $[B/A]^{\searrow X \nearrow X}$ or $[B/A_2]^{\searrow X \nearrow X}$.

Proof. By induction on k.

Corollary 2. *Languages in $\mathcal{L}(mmCDG^{-FA})$ may be not semilinear.*

It should be remarked that the problem of semilinearity is still open for CDG.

The example of L_{exp} suggests that languages in $\mathcal{L}(mmCDG^{-FA})$ may be rather complex. Indeed, we prove that the membership problem in this family is NP-complete.

[11] I.e. the valency projection is reducible to this string of valencies.

Theorem 3. *Membership problem for $mmCDG^{-FA}$ is NP-complete.*

Proof. $[NP - hardness]$. We will reduce the problem of satisfiability of $3-CNF$ to the membership problem for $mmCDG^{-FA}$. For this, we will define an $mmCDG^{-FA}$-grammar $G^{(3)}$ and, for every 3–CNF Φ, we will construct a string $w(\Phi)$ such that $w(\Phi) \in L(G^{(3)})$ iff $\Phi \in SAT$. This is a definition of $G^{(3)}$:
Dictionary: $W = \{*, x, \bar{x}, y, b, f, F\}$.
x corresponds to occurrences of the propositional letters in the clauses of 3–CNF. $\bar{x}$ corresponds to occurrences of the negated propositional letters. y corresponds to the propositional letters which have no occurrences in a clause. The following example explains how the clauses are coded using these symbols. Supposing that there are only seven letters $x_1, \ldots, x_7$, the clause $x_2 \vee \neg x_4 \vee x_7$ is represented by the string $yxy\bar{x}yyx$. For a clause C, $g(C)$ will denote the string representing C.
Elementary types:
$\mathbf{C} = \{S, 0, 1, 0', 1', A, B, T\}$.
Modes:
$\pi(0) = \{1\}$, $\pi(1) = \{0\}$, $\pi(0') = \{1'\}$, $\pi(1') = \{0'\}$. Intuitively, these negative modes mean that the dependencies 0 and 1 (respectively, $0'$ and $1'$) cannot cross.
Lexicon λ:

$$F \mapsto [S],$$
$$b \mapsto \{[\varepsilon]^{\nearrow 0}, [\varepsilon]^{\nearrow 1}\},$$
$$f \mapsto \left\{ \begin{array}{l} [\varepsilon]^{\searrow 0}, \\ [\varepsilon]^{\searrow 0'}, \\ [\varepsilon]^{\searrow 1}, \\ [\varepsilon]^{\searrow 1'} \end{array} \right\} \qquad y \mapsto \left\{ \begin{array}{ll} [A\backslash A]^{\searrow 1 \nearrow 1'}, & [B\backslash B]^{\searrow 1' \nearrow 1}, \\ [A\backslash A]^{\searrow 0 \nearrow 0'}, & [B\backslash B]^{\searrow 0' \nearrow 0}, \\ [A/A]^{\searrow 1 \nearrow 1'}, & [B/B]^{\searrow 1' \nearrow 1}, \\ [A/A]^{\searrow 0 \nearrow 0'}, & [B/B]^{\searrow 0' \nearrow 0}, \\ [A]^{\searrow 1 \nearrow 1'}, & [B]^{\searrow 1' \nearrow 1}, \\ [A]^{\searrow 0 \nearrow 0'}, & [B]^{\searrow 0' \nearrow 0} \end{array} \right\}$$
$$* \mapsto [T\backslash\varepsilon]$$
$$x \mapsto \left\{ \begin{array}{ll} [A\backslash T/A]^{\searrow 1 \nearrow 1'}, & [B\backslash T/B]^{\searrow 1' \nearrow 1}, \\ [A\backslash A]^{\searrow 1 \nearrow 1'}, & [B\backslash B]^{\searrow 1' \nearrow 1}, \\ [A\backslash A]^{\searrow 0 \nearrow 0'}, & [B\backslash B]^{\searrow 0' \nearrow 0}, \\ [A/A]^{\searrow 1 \nearrow 1'}, & [B/B]^{\searrow 1' \nearrow 1}, \\ [A/A]^{\searrow 0 \nearrow 0'}, & [B/B]^{\searrow 0' \nearrow 0}, \\ [A\backslash T]^{\searrow 1 \nearrow 1'}, & [B\backslash T]^{\searrow 1' \nearrow 1}, \\ [A]^{\searrow 1 \nearrow 1'}, & [B]^{\searrow 1' \nearrow 1}, \\ [A]^{\searrow 0 \nearrow 0'}, & [B]^{\searrow 0' \nearrow 0}, \\ [T/A]^{\searrow 1 \nearrow 1'}, & [T/B]^{\searrow 1' \nearrow 1} \end{array} \right\} \quad \bar{x} \mapsto \left\{ \begin{array}{ll} [A\backslash T/A]^{\searrow 0 \nearrow 0'}, & [B\backslash T/B]^{\searrow 0' \nearrow 0}, \\ [A\backslash A]^{\searrow 1 \nearrow 1'}, & [B\backslash B]^{\searrow 1' \nearrow 1}, \\ [A\backslash A]^{\searrow 0 \nearrow 0'}, & [B\backslash B]^{\searrow 0' \nearrow 0}, \\ [A/A]^{\searrow 1 \nearrow 1'}, & [B/B]^{\searrow 1' \nearrow 1}, \\ [A/A]^{\searrow 0 \nearrow 0'}, & [B/B]^{\searrow 0' \nearrow 0}, \\ [A\backslash T]^{\searrow 0 \nearrow 0'}, & [B\backslash T]^{\searrow 0' \nearrow 0}, \\ [A]^{\searrow 1 \nearrow 1'}, & [B]^{\searrow 1' \nearrow 1}, \\ [A]^{\searrow 0 \nearrow 0'}, & [B]^{\searrow 0' \nearrow 0}, \\ [T/A]^{\searrow 0 \nearrow 0'}, & [T/B]^{\searrow 0' \nearrow 0} \end{array} \right\}$$

Let $\Phi = C_1 \wedge \ldots \wedge C_m$ be a $3-CNF$ with propositional letters in $X = \{x_1, \ldots, x_n\}$.
String $w(\Phi)$ encoding Φ:
$w(\Phi) = b^n F g(C_1)^R * g(C_2) * \cdots * g(C_m)^{(R)} * f^n$, where every even member i is $g(C_i)$ and every odd member i is the mirror image of $g(C_i)$.

Lemma 8. *Φ is satisfiable iff $w(\Phi) \in L(G^{(3)})$.*

Proof. 1. If Φ is satisfied by values $x = v_1, \ldots, x_n = v_n$, we assign to every occurrence i of b in $w(\Phi)$ the type $[\varepsilon]^{\nearrow v_i}$. The choice of types for the symbols x, $\bar{x}$ and y depends on the number j of the block $g(C_j)$ in $w(\Phi)$ and on the position

of their occurrence in the block. If j is odd, then the assigned type has argument subtypes A, otherwise it has argument subtypes B. We show the exact choice of types trough an example. Let us suppose that j is odd (the other case is similar). The clause C_j is made true by some literal x_i or $\neg x_i$. In the first case, for the occurrence i of x in $g(C_j)$ is selected the type $[T/A]^{\searrow 1 \nearrow 1'}$ when $i = 1$, the type $[A\backslash T]^{\searrow 1 \nearrow 1'}$ when $i = n$ and $[A\backslash T/A]^{\searrow 1 \nearrow 1'}$ otherwise. In the second case, the corresponding types will be $[T/A]^{\searrow 0 \nearrow 0'}$, $[A\backslash T]^{\searrow 0 \nearrow 0'}$ and $[A\backslash T/A]^{\searrow 0 \nearrow 0'}$. Every other symbol in a position $k \neq i$ will have a type $t^{\searrow v_k \nearrow v'_k}$, where t is one of the basic types $[A]$, $[A\backslash A]$, $[A/A]$. For instance, if $x_1 = 0, x_2 = 1, x_3 = 0$ and $g(C_j) = xy\bar{x}$, then the first x has the type $[A]^{\searrow 0 \nearrow 0'}$, y has the type $[A\backslash A]^{\searrow 1 \nearrow 1'}$ and the second x has the type $[A\backslash T]^{\searrow 0 \nearrow 0'}$. Finally, $[S]$ is assigned to F, $[T\backslash\varepsilon]$ is assigned to $*$ and for every $1 \leq i \leq n$, if m is odd, then for the occurrence i of f we choose the type $[\varepsilon]^{\searrow v'_i}$, otherwise we choose the type $[\varepsilon]^{\searrow v_{n-i+1}}$. Let us denote by $\Gamma(\Phi, v_1, \ldots, v_n)$ the string of types assigned in this manner to the string $w(\Phi)$ and by $\Gamma(C_j, v_1, \ldots, v_n)$ the substring of types assigned to $g(C_i)$. It is not difficult to see that $\Gamma(\Phi, v_1, \ldots, v_n) \vdash S$. Indeed, the potentials of the types assigned to n consecutive occurrences of b send the positive valencies encoding the values v_i of the corresponding propositional letters x_i. These valencies are intercepted by the dual negative valencies of the types chosen for the letters x, $\bar{x}$ and y in the closest block $g(C_1)$. Due to the definition of the modes π, the discontinuous dependencies 0 do not cross the discontinuous dependencies 1. This means that the valencies will be intercepted in the inverse order. Then every letter in $g(C_1)$ receiving the valency v will send on the positive valency v' and again, because the discontinuous dependencies $0'$ do not cross the discontinuous dependencies $1'$, the valencies will be intercepted in the inverse order (i.e. in the original order of the symbols b), etc. till the symbols f. This means that all valencies will be paired and eliminated. Besides this, $\Gamma(C_j, v_1, \ldots, v_n) \vdash [T]^{P_j}$ for some P_j and for every $1 \leq j \leq m$. Finally, every type $[T]$ is eliminated by the type assigned to the corresponding occurrence of $*$. So by Theorem 1, $\Gamma(\Phi, v_1, \ldots, v_n) \vdash S$.

2. Let Φ be not satisfiable. Let us assume that $w(\Phi) \in L(G^{(3)})$. Then the types assigned to the occurrences of symbols b correspond to the assignments of the coded values to the corresponding propositional letters. Let us suppose that C_{j_0} is the first false clause in Φ. Without loss of generality, we can suppose that j_0 is odd. Let Γ be a string of types assigned to $w(\Phi)$ in $G^{(3)}$ and Γ_{j_0} be its substring of types assigned to $g(C_{j_0})^R*$. Then Γ_{j_0} must reduce to $[\varepsilon]^{P_{j_0}}$ for some potential P_{j_0}. Therefore, there should be an occurrence i of one of the symbols x or $\bar{x}$ to which is assigned a type with the head dependency T, for instance the type $[A\backslash T/A]^{\searrow 1 \nearrow 1'}$ (or respectively $[A\backslash T/A]^{\searrow 0 \nearrow 0'}$). But, to be eliminated, this type needs paring the valency $\searrow 1$ (respectively $\searrow 0$), which is impossible, because C_{j_0} is false, so the value of x_i is dual to the choice of valencies. This means that the potential in Γ cannot be balanced. Hence, $w(\Phi) \notin L(G(\phi))$.

[$NP-completeness$]. Let us define the following relation $\prec$ on the set of discontinuous dependencies (i.e. pairs of the form $a(d) = \nearrow d \searrow d$ or $a(d) = \swarrow d \nwarrow d$): $a(d_1) \prec_\pi a(d_2)$ if the two dependencies cross and $d_1 \in \pi(d_2)$. With respect to

the pairing rule **FA**, this relation means that $a(d_1)$ must be paired before $a(d_2)$. More precisely, the following proposition holds:

Lemma 9. *A potential P is balanced with respect to $\mathbf{FA}^{l}_{C:\pi(C)}$ with negative modalities π iff there is such pairing of valencies in P that $\prec_\pi$ has no circles on crossing dependencies.*

So the nondeterministic polynomial algorithm for membership $w \in L(G)$ is as follows:
1) guess $\Gamma \in \lambda(w)$,
2) check $\|\Gamma\|_l \vdash_c [S]$,
3) guess a pairing of valencies in $\|\Gamma\|^*_v$,
4) check that it is balanced and
5) check that $\prec_\pi$ has no circles on crossing dependencies.

By the way, using these techniques we can define a $mmCDG^{-FA}$ generating the language $\{ww^Rw \mid \ w \in \{a,b\}^+\}$ and, for every Turing machine, define a $mmCDG^{-FA}$ generating the "protocols" of its computations. We can also generate the copy language L_{copy} using the pairing rule $\mathbf{FC}_\pi$ with negative modes.

6 Conclusion

This study shows that the class of $mmCDG^{-FA}$-grammars may serve as a general theoretical frame for categorial dependency grammars, in which the languages form an AFL, may be not semilinear and the membership problem is NP-complete. Meanwhile, its subset of polynomially parsed $mmCDG$ without negative modes is perfectly adequate and sufficient for practical use because in the text corpora and more generally, in the written speech the sentences are explicitly separated by punctuation markers. So they are analyzed independently. As to the iterated constructions in the sentences, they are immediately definable through the primitive iterated types. On the other hand, for this subfamily $mmCDG$, the problems of semilinearity, of closure under iteration and of inclusion $L_{copy} \in \mathcal{L}(mmCDG)$ still rest open.

The two main lessons of this study are that:
(a) checking individual long distance discontinuous dependencies is a polynomial time task, whereas checking interaction of at least four of them may be untractable;
(b) for categorial dependency grammars, their closure under Kleene iteration may be obtained by prohibition of crossing one selected discontinuous dependency.

7 Acknowledgements

We are grateful to an anonymous reviewer who checked the proofs and pointed out several imprecisions and misprints.

References

1. Y. Bar-Hillel, H. Gaifman, and E. Shamir. On categorial and phrase structure grammars. *Bull. Res. Council Israel*, 9F:1–16, 1960.
2. Denis Béchet, Alexander Dikovsky, and Annie Foret. Dependency structure grammars. In P. Blache and E. Stabler, editors, *Proc. of the 5th Int. Conf. "Logical Aspects of Computational Linguistics" (LACL'2005)*, LNAI 3492, pages 18–34, 2005.
3. M. Dekhtyar and A. Dikovsky. Generalized categorial dependency grammars. In A. Aviron et al., editor, *Trakhtenbrot/Festschrift*, LNCS 4800, pages 230–255. Springer Verlag, 2008.
4. Michael Dekhtyar and Alexander Dikovsky. Categorial dependency grammars. In M. Moortgat and V. Prince, editors, *Proc. of Intern. Conf. on Categorial Grammars*, pages 76–91, Montpellier, 2004.
5. A. Dikovsky. Multimodal categorial dependency grammars. In *Proc. of the 12th Conference on Formal Grammar*, pages 1–12, Dublin, Ireland, 2007.
6. Alexander Dikovsky. Dependencies as categories. In D. Duchier and G.-J. M. Kruijff, editors, *"Recent Advances in Dependency Grammars". COLING'04 Workshop*, pages 90–97, 2004.
7. Denis Duchier and Ralf Debusmann. Topological dependency trees: A constraint-based account of linear precedence. In *Proc. of the 39th Intern. Conf. ACL'2001*, pages 180–187. ACL & Morgan Kaufman, 2001.
8. Richard A. Hudson. *Word Grammar*. Basil Blackwell, Oxford-New York, 1984.
9. Aravind K. Joshi, Vijay K. Shanker, and David J. Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S. Shieber, and T. Wasow, editors, *Foundational issues in natural language processing*, pages 31–81, Cambridge, MA, 1991. MIT Press.
10. B. N. Karlov. Normal forms and automata for categorial dependency grammars. *Vestnik Tverskogo Gosudarstvennogo Universiteta (Annals of Tver State University). Series: Applied Mathematics*, 35 (95):23–43, 2008. (in Russ.).
11. J. Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of languages and its mathematical aspects*, pages 166–178. American Mathematical Society, Providence RI, 1961.
12. I. Mel'čuk. *Dependency Syntax*. SUNY Press, Albany, NY, 1988.
13. Michael Moortgat and Glin V. Morrill. Heads and phrases. Type calculus for dependency and constituent structure. Ms OTS, Utrecht, 1991.
14. G. V. Morrill. *Type Logical Grammar. Categorial Logic of Signs*. Kluwer, Dordrecht, 1994.
15. D. Sleator and D. Temperly. Parsing English with a Link Grammar. In *Proc. IWPT'93*, pages 277–291, 1993.
16. M. Steedman and J. Baldridge. Combinatory categorial grammar. In K. Brown, editor, *Encyclopedia of Language and Linguistics. Vol 2*, pages 610–622. Elsevier, Oxford, 2006.
17. L. Tesnière. *Éléments de syntaxe structurale*. Librairie C. Klincksieck, Paris, 1959.
18. K. Vijay-Shanker and D.J. Weir. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–545, 1994.