# Multi-push-down Languages and Grammars*

Alessandra Cherubini[1] Luca Breveglieri[2]
Claudio Citrini[1] Stefano Crespi Reghizzi[2]

[1]Dipartimento di Matematica (DM)
[2]Dipartimento di Elettronica e Informazione (DEI)
Politecnico di Milano (PdM)
Piazza Leonardo Da Vinci n° 32
I-20133 Milano, Italy
Ph: +39 (0)2 2399 1 (switchboard) - Fax: +39 (0)2 2399 4568[1] 3411[2]
E-mail: aleche,clacit@mate.polimi.it, brevegli,crespi@elet.polimi.it

**Abstract.** *A new class of languages, called multi-push-down (mpd), that generalize the classical context-free (cf, or Chomsky type 2) ones is introduced. These languages preserve some important properties of cf languages: a generalization of the Chomsky-Schützenberger homomorphic characterization theorem, the Parikh theorem and a "pumping lemma" are proved. Multi-push-down languages are an AFL. Their recognizers are automata equipped with a multi-push-down tape. Multi-push-down languages form a hierarchy based on the number of push-down tapes.*

## 1 Introduction

This research studies a new class of languages, called multi-push-down (mpd), that generalize the classical context-free (cf, or Chomsky type 2) ones [14], taking a new direction. We hasten to say that this generalization has nothing to do with past proposals to increase the generative capacity of type 2 grammars by introducing some sort of context-dependency, as for instance in matrix grammars [12, 20] or in other regulated rewriting systems [12]. Our approach is based on a different and more powerful operator for combining the constituents occurring in the right-hand side of a production, which replaces the simple catenation operation used by type 2 rules. Apart from this difference, the productions of the mpd grammars behave as the usual ones, a fact having the desirable consequence that the important properties of cf grammars are preserved or generalized by mpd grammars: it is so for the Chomsky-Schützenberger theorem [11], for the Parikh theorem [19], for the "pumping lemma" [17] and for a few more properties. The derivations of mpd grammars can be represented by means of syntax trees, with a suitable order of visit. Considering the generative capacity, mpd languages include some well-known non-context-free languages: homomorphic replications [13] provide an abstract example; the nested procedure declarations of the Ada

* Work supported by a grant of the "Ministero dell'Università e della Ricerca Scientifica e Tecnologica" (60%), Italy, and by ESPRIT BRA ASMICS 2, Special Contract n° 6317, European Union.

programming language are a practical case. Such languages exhibit the long-range dependencies that have always embarrassed formal linguists.

Finally, a word on recognition and parsing, to complete the picture: mpd languages can be parsed in polynomial time, as proved in a related paper by A. Cherubini and P. Sanpietro [10], who have extended the Cocke-Kasami-Younger algorithm [14].

After this quick survey of the niceties of mpd grammars the rest of the introduction intuitively presents their generative mechanism. Perhaps the best way is to start from a cf grammar in Greibach normal form [20], with productions of the form $A \rightarrow bA_1A_2 \ldots A_n$. It is well known that this production can be given the following interpretation as an instruction for a push-down (pd) automaton: if $A$ is the top symbol of the pd store (a LIFO data structure), pop it upon reading $b$ from the input and push the string $A_1A_2 \ldots A_n$ onto the store. The machine starts with the axiom in the store and recognizes by empty store. Notice also that this machine is stateless and in general non-deterministic. We recall that for cf grammars derivations can be assumed to be leftmost, without loss of generality, and that the previous automaton operates in the leftmost manner.

Multi-push-down grammars are organized as an infinite hierarchy indexed by a parameter $n \geq 1$, with the case $n = 1$ coinciding with cf grammars. Take for simplicity $n = 2$; a production of a grammar corresponding to a 2-pd automaton takes the form: $A \rightarrow b(A_1A_2 \ldots A_h)(B_1B_2 \ldots B_k)$, and can be interpreted as an instruction for a store organized as the concatenation of two pd tapes (see Figure 1), in essentially the same manner as the production in Greibach normal form for one pd tape. The interpretation is then the following: if $A$ is the top symbol of the store, pop it upon reading $b$ from the input, push the string $A_1A_2 \ldots A_h$ onto the first pd tape of the store and push the string $B_1B_2 \ldots B_k$ onto the second pd tape of the store. Notice that the symbol $A$ is popped from the first pd tape unless it is empty; in this last case $A$ is popped from the second pd tape. Also this machine is stateless, starts with the axiom in the store and recognizes by empty store; in general, the machine is non-deterministic, too. In the $n$-pd case, the store is made of $n \geq 1$ adjacent pd tapes, which are linearly ordered; the reading head of the automaton is allowed to pop one symbol of the store alphabet from a pd tape only if the preceding pd tapes are empty. Instead, all pd tapes can be written in parallel by one move, regardless whether they are empty or not. The straightforward correspondence between the generative grammar and the automaton is thus extended from cf to mpd languages.

Notice that the above automata are stateless, since all information is stored in the pd tapes of the automaton. The family of stateless, in general non-deterministic, mpd automata and the family of mpd grammars are equivalent, as mentioned above. Finite states are not required as long as the purpose is the equivalence between mpd automata and grammars. However, finite states are allowed in mpd automata, but do not increase the recognition power of such a family of automata, as long as non-determinism is allowed; in fact, if non-determinism is tolerated any mpd automaton, with finite states, admits an equivalent stateless mpd automaton.

This fact extends a well-known property of pd automata [14]. Moreover, the analogy between the cf and mpd cases goes farther, since the family of $n$-pd automata recognizing by final state is as powerful as the family of $n$-pd automata recognizing by empty store, for any fixed $n \geq 1$.

The addition of the determinism constraint reduces the recognition power of mpd automata, for a fixed number $n$ of pd tapes. For instance, this happens for $n = 1$, because 1-pd automata are push-down automata and it is well-known that deterministic push-down automata are less powerful than non-deterministic ones [14]. In the conclusion this point is resumed.

An additional comparison may help in understanding the proposed model, before entering the technical presentation. A 2-pd automaton is very different from a machine with two independent pd tapes. The latter can simulate a Turing machine by storing its semitapes on the pd tapes [15]. In our case this is not possible because only one tape at a time can be read, so that the information written on, say, the right semitape would not be accessible until the left semitape is emptied. As a consequence mpd languages are much less general than context-sensitive ones, in fact they are permutations of cf languages.

Section 2 defines mpd grammars and automata, presents some illustrative examples, introduces a normal form and proves the equivalence of mpd grammars and automata. Section 3 first proves the central properties of each $n$-pd family for any fixed $n \geq 1$, then it proves closure and inclusion properties of the whole hierarchy. The Conclusion discusses determinism, parsing and points to a related investigation on grammars which further generalize cf ones by having a store made of FIFO as well as of LIFO tapes. The Appendices A and B present two lengthy proofs.

## 2 Definitions and examples

This section defines the *multi-push-down* (mpd) automata with $n \geq 1$ pd tapes (also called $n$-pd or $PD^n$ automata) and the corresponding equivalent class of grammars, the *multi-depth* grammars (also called depth-$n$ or $D^n$ grammars). A normal form for such grammars is introduced. Some examples of the recognitive and generative power of the families of $n$-pd automata and depth-$n$ grammars are provided.

The mpd automaton, shown in Figure 1 for the case $n = 2$, has one read-only left to right input tape and $n \geq 1$ read-write memory tapes with a LIFO rewriting policy. The machine performs the following actions with one move:

- reads one or zero symbols from the input tape and moves past the read symbol;
- reads the symbol on the top of the first pd tape; if the first pd tape is empty it reads the top symbol of the second non-empty pd tape; and so on.
- switches its internal state;
- possibly writes in parallel $n$ finite strings $\alpha_i$ on the $i$-th pd tape, with respectively $i = 1, 2, \ldots, n$. The $i$-th head moves to the left of the inserted string $\alpha_i$, i.e. writing is a push move.

**Fig. 1.** *A 2-pd automaton $M$.*

The next definition is the same as the classical definition of pd automaton, apart from the fact that our machine can write in parallel into $n$ pd tapes instead of just one.

**Definition 1.** A *$n$-push-down* ($n$-pd or $PD^n$) *automaton* $M$, with $n \geq 1$, accepting by final state, is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F, Z_0)$, where:

- $Q$ is a finite non-empty set of internal states
- $\Sigma$ (input) and $\Gamma$ (memory) are finite alphabets
- $\delta$ is a partial transition mapping

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \Rightarrow \wp_F (Q \times (\Gamma^*)^n)$$

  where $\wp_F(E)$ is the set of the finite subsets of a set $E$

– $q_0 \in Q$ is the initial state

– $F \subseteq Q$ is the set of final states

– $Z_0 \in \Gamma$ is the initial memory symbol

A *configuration* of $M$ is a $(n+2)$-tuple $\langle q, x; \gamma_1, \ldots, \gamma_n \rangle$, where $q \in Q$, $x \in \Sigma^*$ and $\gamma_1, \ldots, \gamma_n \in \Gamma^*$. The configuration $\langle q_0, x; Z_0, \varepsilon, \ldots, \varepsilon \rangle$ is *initial*. A configuration $\langle q, x; \gamma_1, \ldots, \gamma_n \rangle$, where $q \in F$, is called *final*.

The *transition relation* $\overset{*}{\vdash}_M$ is the transitive closure of the binary relation $\vdash_M$, over configurations, defined in the following way:

$$\langle q, ax; \varepsilon, \ldots, \varepsilon, A\gamma_i, \ldots, \gamma_n \rangle \vdash_M \langle q', x; \alpha_1, \ldots, \alpha_{i-1}, \alpha_i\gamma_i, \ldots, \alpha_n\gamma_n \rangle$$

if $(q', \alpha_1, \ldots, \alpha_n) \in \delta(q, a, A)$, where $a \in \Sigma \cup \{\varepsilon\}$, for some $1 \leq i \leq n$.

In words, the move $(q', \alpha_1, \ldots, \alpha_n) \in \delta(q, a, A)$ reads "$a$" or "$\varepsilon$" from the input, reads "$A$" from the head of the store (in the first non-empty segment), then switches the state to "$q'$" and for all $j$, with $1 \leq j \leq n$, writes $\alpha_j$ by means of the $j$-th writing head. Accordingly, the $j$-th writing head moves leftwards.

A string $x$ is *accepted* (by final state) by a $n$-pd automaton $M$ if and only if

$$\langle q_0, x; Z_0, \varepsilon, \ldots, \varepsilon \rangle \overset{*}{\vdash}_M \langle q, \varepsilon; \gamma_1, \ldots, \gamma_n \rangle$$

where $q \in F$. □

Clearly a 1-pd machine coincides with a classical pd automaton.

**Definition 2.** The family of languages recognized by $n$-pd automata is denoted $\mathcal{L}_{PD^n}$, for any $n \geq 1$. □

*Example* 2.1 The non-cf language $\{a^n b^n c^n \mid \quad n \geq 0\} \in \mathcal{L}_{PD^2}$. An accepting non-deterministic 2-pd automaton is $M = (Q, \Sigma, \Gamma, \delta, q_0, \{q_2\}, Z_0)$, where:

$$\Sigma = \{a, b, c\} \quad \Gamma = \{Z_0, B, C, D\} \quad Q = \{q_0, q_1, q_2\}$$

$$\delta(q_0, a, Z_0) = \{(q_1, Z_0B, CD)\}$$

$$\delta(q_0, \varepsilon, Z_0) = \{(q_2, \varepsilon, \varepsilon)\}$$

$$\delta(q_1, a, Z_0) = \{(q_1, Z_0B, C), (q_1, B, C)\}$$

$$\delta(q_1, b, B) = \{(q_1, \varepsilon, \varepsilon)\}$$

$$\delta(q_1, c, C) = \{(q_1, \varepsilon, \varepsilon)\}$$

$$\delta(q_1, \varepsilon, D) = \{(q_2, \varepsilon, \varepsilon)\}$$

The automaton $M$ reads the $a$'s in the initial state while $Z_0$ is on the top of the first pd tape, switches to the state $q_1$ and writes $Z_0B$ and $CD$ onto the first

and the second pd tape, respectively. Then in the state $q_1$ it reads the $a$'s from the input tape while $Z_0$ is on the top of the first pd tape, storing the $a$'s as $B$'s and $C$'s onto the first and the second pd tape, respectively. When the $a$'s are finished and $Z_0$ disappears from the top of the first pd tape, $M$ reads from the input tape a number of $b$'s equal to the number of $B$'s. When the first pd tape is empty, the automaton consumes the $C$'s from the second pd tape while reading the $c$'s from the input tape. When $D$ is the top symbol of the store, with an $\varepsilon$-move $M$ reaches the final state $q_2$ and recognizes.

We have defined acceptance by final state, but, analogously to the 1-pd, or cf, case, acceptance by empty store could be defined as well. In this case the set $F$ of final states is not defined and the final configurations are of the type $\langle q, x; \varepsilon, \ldots, \varepsilon \rangle$, for any $q \in Q$. Furthermore, the following statement holds.

**Statement 3.** A mpd language $L$ is recognized by a non-deterministic $n$-pd automaton by empty store if and only if there exists a non-deterministic $n$-pd automaton recognizing $L$ by final state, for some $n \geq 1$. □

The proof, a simple generalization of the well-known proof for pd automata [14], is omitted.

The previous statement permits to define a generative system for mpd languages, i.e. the grammars, called depth-$n$ or $D^n$ grammars[2], generating the languages in $\mathcal{L}_{PD^n}$. A $D^n$ *grammar* is a rewriting system that rewrites one non-terminal symbol $A$ occurring in a string $\beta$ by a string $\gamma$. Unlike cf grammars, $\gamma$ need not be written contiguously as a replacement of $A$; instead, $\gamma$ is a list of $n \geq 1$ strings, $\gamma = (\gamma_1)(\gamma_2)\ldots(\gamma_n)$, and each string $\gamma_i$ is inserted in a marked position of $\beta$. First we need define lists.

**Definition 4.** Let $\Sigma$ be a finite alphabet and let "(", ")" be characters not in $\Sigma$. A *list* is a finite sequence of (possibly empty) strings enclosed by the symbols "(", ")", i.e. $\overline{\gamma} = (\gamma_1)(\gamma_2)\ldots(\gamma_n)$, with $n \geq 1$, where $\gamma_i \in \Sigma^*$, for any $1 \leq i \leq n$. The string $\gamma_i$ is called the $i$-th component of the list $\overline{\gamma}$. Lists are elements of $(\text{"("}\Sigma^*\text{")"})^+$. List names will be overscored, to distinguish them from string names. A list with $n$ components is called a *list of degree* $n$, or a $n$-list. An equivalent notation for a $n$-list is $(\gamma_1)_1(\gamma_2)_2\ldots(\gamma_n)_n$, which allows to drop the empty components.

A list can be transformed into a string in the natural way by the homomorphism (*unmarking*) $u : (\Sigma \cup \{\text{"("}\} \cup \{\text{")"}\})^* \to \Sigma^*$, defined by $u(\text{"("}) = \varepsilon$, $u(\text{")"}) = \varepsilon$ and $u(a) = a$, for any $a \in \Sigma$. Usually we denote $\gamma = u(\overline{\gamma})$, deleting the overscore, and we call the string $\gamma$ the *unmarked copy* of the list $\overline{\gamma}$. □

**Definition 5.** A $D^n$ *grammar* $G$ is a 4-tuple $G = (V_N, V_T, P, S)$, where $V_N$ and $V_T$ are the non-terminal and terminal alphabet, respectively, $S \in V_N$ is the axiom and $P$ is a finite set of elements (productions) of the form

$$A \to w(\alpha_1)_1(\alpha_2)_2\ldots(\alpha_n)_n$$

[2] The letter $D$ stands for "depth-first", in contrast to the "breadth-first" grammars of [1, 8, 9].

where $w \in V_T^*$ and $\alpha_i \in V_N^*$, for any $1 \leq i \leq n$. The string $\alpha_i$ is called the $i$-th component of the production. $\square$

Notice that a production can be equivalently written as $A \rightarrow w\overline{\alpha}$, where $\overline{\alpha} = (\alpha_1)(\alpha_2)\ldots(\alpha_n)$ is a $n$-list. The right-hand side $w\overline{\alpha}$ is a $n$-list over the alphabet $V_N$ with a prefix in $V_T^*$. We continue to call such entities $n$-lists.

For brevity the empty components of a list can be shortened as follows:

$A \rightarrow w(\varepsilon)(\varepsilon)\ldots(\varepsilon)$ becomes $A \rightarrow w$

$A \rightarrow (\varepsilon)(\varepsilon)\ldots(\varepsilon)$ becomes $A \rightarrow \varepsilon$

$A \rightarrow w(\varepsilon)_1\ldots(\varepsilon)_{i-1}(\alpha_i)_i(\varepsilon)_{i+1}\ldots(\varepsilon)_n$ becomes $A \rightarrow w(\alpha_i)_i$

Obviously for $n = 1$ a $D^n$ grammar is cf.

A derivation is a relation between two $n$-lists, such that the latter one is obtained by rewriting the leftmost non-terminal of the former one using a production.

**Definition 6.** Let $\overline{\beta} = (\varepsilon)_1\ldots(\varepsilon)_{i-1}(A\beta)_i(\beta_{i+1})_{i+1}\ldots(\beta_n)_n$ be a $n$-list, for some $1 \leq i \leq n$, where $\beta_j \in V_N^*$, for each $i \leq j \leq n$, and $A \in V_N$. Take a string $x \in V_T^*$. We write the following derivation

$$x\overline{\beta} \Rightarrow xw(\alpha_1)\ldots(\alpha_{i-1})(\eta_i)(\eta_{i+1})\ldots(\eta_n)$$

if $A \rightarrow w(\alpha_1)_1(\alpha_2)_2\ldots(\alpha_n)_n$ is a production and $\eta_j = \alpha_j\beta_j$, for every $j$ with $i \leq j \leq n$. $\square$

Notice that only *leftmost* derivations are defined. In fact, by relaxing the leftmost constraint on the order of derivation the generative power of $D^n$ grammars increases, for any $n \geq 2$, in contrast with the behaviour of cf grammars (remember however that cf grammars coincide with $D^n$ grammars for $n = 1$). As usual $\stackrel{*}{\Rightarrow}$ denotes the reflexive and transitive closure of the relation $\Rightarrow$.

The generation of a string starts with the list $(S)_1(\varepsilon)_2\ldots(\varepsilon)_n$. A terminal string $x$ is *derivable* from $S$ if and only if $(S)_1(\varepsilon)_2\ldots(\varepsilon)_n \stackrel{*}{\Rightarrow} x(\varepsilon)\ldots(\varepsilon)$ or, dropping the $\varepsilon$'s, $(S)_1 \stackrel{*}{\Rightarrow} x$, or also, for brevity, $S \stackrel{*}{\Rightarrow} x$.

The language generated by a $D^n$ grammar $G$ is $L(G) = \{x \in V_T^* | \quad S \stackrel{*}{\Rightarrow}_G x\}$.

**Definition 7.** $\mathcal{G}_{D^n}$ is the family of $D^n$ grammars; the corresponding family of languages is denoted by $\mathcal{L}_{D^n}$ and is named the family of depth-$n$ languages. $\square$

The following examples highlight the generative capacity of $D^n$ grammars, giving also detailed examples of derivations.

*Example* 2.2 A $D^2$ grammar $G$ generating the non-cf language $\{a^nb^nc^n | \quad n \geq 0\}$ of example 2.1 is $G = (V_N, V_T, P, S)$:

$$V_N = \{S, B, C\} \quad V_T = \{a, b, c\} \text{ and } P = \begin{cases} S \rightarrow a(SB)_1(C)_2 \quad | \quad \varepsilon \\ B \rightarrow b \\ C \rightarrow c \end{cases}$$

**Fig. 2.** *Syntax tree of the string "aabbcc".*

Figure 2 shows the syntax tree of the above derivation of the string *aabbcc*. The thick edges point out the rewritings between different list components.

*Example* 2.3 The non-cf language $\{ww| \quad w \in \{a,b\}^*\}$ (COPY language, see [3]) is in $\mathcal{L}_{D^2}$. A $D^2$ grammar generating this language is $G = (V_N, V_T, P, S)$, with:

$$V_N = \{S, A, B, A', B'\} \quad V_T = \{a, b\}$$

$$P = \begin{cases} S \rightarrow a(SA')_1 \quad | \quad b(SB')_1 \quad | \quad \varepsilon \\ A' \rightarrow (A)_2 \\ B' \rightarrow (B)_2 \\ A \rightarrow a \\ B \rightarrow b \end{cases}$$

An example of derivation of the string *abbabb* is:

$$S \Rightarrow a(SA')_1 \Rightarrow ab(SB'A')_1 \Rightarrow abb(SB'B'A')_1$$

$$\Rightarrow abb(B'B'A')_1 \Rightarrow abb(B'A')_1(B)_2 \Rightarrow abb(A')_1(BB)_2$$

$$\Rightarrow abb(ABB)_2 \Rightarrow abba(BB)_2 \Rightarrow abbab(B)_2 \Rightarrow abbabb$$

The grammar $G$ works by first storing a reverse copy of $w$ onto the first component of the list and then by reversing it again onto the second component.

*Example* 2.4 As a practical example in $\mathcal{L}_{D^2}$, we consider the language of nested procedure declarations, with procedure identifiers repeated after the procedure "end" (as in the programming language Ada). Procedure identifiers need not be unique (for instance "proc abc proc abc end abc end abc" is accepted). A $D^2$ grammar for the language is $G_1 = (V_N, V_T, S_0, P) \in \mathcal{G}_{D\,2}$:

$$V_N = \{S_0, S_1, S_2, T_1, A, \ldots, Z, A', \ldots, Z'\} \quad V_T = \{\langle proc\rangle, \langle end\rangle, a, \ldots, z\}$$

$$P = \begin{cases} S_0 \rightarrow \langle proc\rangle(S_1S_2)_1(S_0)_2 \quad | \quad \varepsilon \\ S_1 \rightarrow \varepsilon \quad | \quad a(S_1A)_1 \quad | \quad b(S_1B)_1 \quad | \quad \ldots \quad | \quad z(S_1Z)_1 \\ A \rightarrow (A')_2 \\ \ldots\ldots\ldots \\ Z \rightarrow (Z')_2 \\ A' \rightarrow a \\ \ldots\ldots \\ Z' \rightarrow z \\ S_2 \rightarrow \langle proc\rangle(S_1S_2)_1(T_1)_2 \quad | \quad (T_1)_2 \\ T_1 \rightarrow \langle proc\rangle(S_1S_2)_1 \quad | \quad \langle end\rangle \end{cases}$$

**Definition 8.** We also need a notation for the multi-depth set or *superfamily*:

$$\mathcal{G}_{D^+} = \{G| \quad \forall n \geq 1 \; G \in \mathcal{G}_{D^n}\} \qquad \text{and} \qquad \mathcal{L}_{D^+} = \{L| \quad \forall n \geq 1 \; L \in \mathcal{L}_{D^n}\}$$

□

Normal forms of $D^n$ grammars In order to simplify several proofs, we introduce in two steps a form similar to the Chomsky normal form of cf grammars [14].

**Definition 9.** A $D^n$ grammar $G = (V_N, V_T, P, S)$ is in *separate normal form* if the productions are of the following types:

| | |
|---|---|
| $A \rightarrow (\alpha)_i$ | where $\alpha \neq \varepsilon$ and $1 \leq i \leq n$ |
| $A \rightarrow w$ | where $w \in V_T^*$ |
| $S \rightarrow \varepsilon$ | if and only if $\varepsilon \in L(G)$ |

□

In Statement 22 it is proved that the membership problem of the empty string $\varepsilon$ is decidable for depth-$n$ languages.

**Statement 10.** For each grammar $G \in \mathcal{G}_{D^n}$ there exists an equivalent grammar $G' \in \mathcal{G}_{D^n}$ in separate normal form. □

**Proof** First notice that it is possible to avoid the productions $A \to \varepsilon$ with $A \neq S$. In fact, analogously to the cf case, any production $A \to \varepsilon$ can be deleted by adding to each production with $A$ on the right hand side the production(s) obtained by deleting $A$ (in all ways) from the string on the right hand side. Now we construct a new grammar $G' = (V'_N, V_T, P', S)$ from $G$, as follows:

$$V'_N = V_N \cup V''_N \cup V'''_N$$

where

$$V''_N = \{\langle w \rangle | \quad A \to w(\alpha_1)_1 \dots (\alpha_n)_n \text{ is a production of } G\}$$

$$V'''_N = \{\langle \alpha_i{}^{(i)} \rangle | \quad \alpha_i \neq \varepsilon \text{ is the } i\text{−th component of a production of } G\}$$

For each production

$$A \to w(\alpha_1)_1 \dots (\alpha_n)_n \in P \tag{1}$$

with $\alpha_j \in V_N^*$ for any $1 \leq j \leq n$, construct the production

$$A \to (\langle w \rangle \langle \alpha_n^{(n)} \rangle \dots \langle \alpha_1^{(1)} \rangle)_1$$

where $\langle \alpha_j^{(j)} \rangle$ is empty if $\alpha_j = \varepsilon$. In addition, construct the productions:

$$\langle w \rangle \to w \quad \text{and} \quad \langle \alpha_j^{(j)} \rangle \to (\alpha_j)_j$$

Clearly the grammar $G'$ is in separate normal form. We show that $L(G) \subseteq L(G')$. Consider a list

$$\overline{\beta} = (\varepsilon)_1 \dots (\varepsilon)_{i-1} (A\beta_i)_i (\beta_{i+1})_{i+1} \dots (\beta_n)_n \tag{2}$$

for some $1 \leq i \leq n$, and a derivation

$$\overline{\beta} \Rightarrow_G w(\alpha_1)_1 \dots (\alpha_{i-1})_{i-1} (\eta_i)_i (\eta_{i+1})_{i+1} \dots (\eta_n)_n = \overline{\gamma} \tag{3}$$

where $\eta_j = \alpha_j \beta_j$, for any $i \leq j \leq n$, via the production (1). Then

$$\overline{\beta} \Rightarrow_{G'} (\langle w \rangle \langle \alpha_n^{(n)} \rangle \dots \langle \alpha_1^{(1)} \rangle)_1 (\beta_i)_i (\beta_{i+1})_{i+1} \dots (\beta_n)_n$$

$$\Rightarrow_{G'} w(\langle \alpha_n^{(n)} \rangle \dots \langle \alpha_1^{(1)} \rangle)_1 (\beta_i)_i (\beta_{i+1})_{i+1} \dots (\beta_n)_n$$

$$\Rightarrow_{G'} w(\langle \alpha_{n-1}{}^{(n-1)} \rangle \dots \langle \alpha_1^{(1)} \rangle)_1 (\beta_i)_i (\beta_{i+1})_{i+1} \dots (\eta_n)_n$$

$$\Rightarrow_{G'} \dots \Rightarrow_{G'} \overline{\gamma}$$

Thus $L(G') \supseteq L(G)$. We omit the proof that $L(G) \supseteq L(G')$, which is alike. Hence it follows $L(G) = L(G')$.

□

The $D^n$ grammars with $n \geq 2$ also admit more refined normal forms.

**Definition 11.** A $D^n$ grammar $G = (V_N, V_T, P, S)$, for $n \geq 2$, is in *binary normal form* if its productions are of the following types:

$A \rightarrow (\alpha)_1$ where $1 \leq |\alpha| \leq 2$

$A \rightarrow (B)_i$ where $B \in V_N$ and $1 < i \leq n$

$A \rightarrow a$ where $a \in V_T$

$S \rightarrow \varepsilon$ if and only if $\varepsilon \in L(G)$

□

**Statement 12.** For each grammar $G \in \mathcal{G}_{D^n}$, with $n \geq 2$, there exists an equivalent grammar $G' \in \mathcal{G}_{D^n}$ in binary normal form. □

**Proof** From Statement 10 we assume that $G$ is in separate normal form, and we construct a grammar $G'$ in binary normal form. If $G$ violates the binary normal form only because $P$ includes productions of the type

$$A \rightarrow w \quad \text{with } w \in V_T^+ \text{ and } |w| > 1$$

we can proceed similarly to the classical Chomsky normal form and the proof is omitted. For the productions not in binary normal form we have to study two cases:

1. $A \rightarrow (\alpha)_1$ with $|\alpha| > 2$
2. $A \rightarrow (\alpha)_i$ with $|\alpha| \geq 2$ for some $1 < i \leq n$

First we construct the productions of $G'$ in each case, then we prove the equivalence. For simplicity we shall assume $\alpha_1, \alpha_i = XYZ$ (the construction is easily generalized, and the proof is made by induction).
*Case* (1). We create for $G'$ the new non-terminal $\langle XY \rangle^{(1)}$ and we replace the production $A \rightarrow (\alpha)_1$ by

$$A \rightarrow (\langle XY \rangle^{(1)} Z)_1 \quad \text{and} \quad \langle XY \rangle^{(1)} \rightarrow (XY)_1$$

*Case* (2). It requires a little change in the order. We have

$$A \rightarrow (\alpha)_i \quad \text{with } |\alpha| \geq 2 \text{ for some } 1 < i \leq n$$

Then we create for $G'$ the new non-terminals $\langle ZY \rangle^{(i)}$, $\langle Z \rangle^{(i)}$, $\langle Y \rangle^{(i)}$ and $\langle X \rangle^{(i)}$, and we replace the production $A \rightarrow (\alpha)_i$ by

$$A \rightarrow (\langle ZY \rangle^{(i)} \langle X \rangle^{(i)})_1 \quad \text{and} \quad \langle ZY \rangle^{(i)} \rightarrow (\langle Z \rangle^{(i)} \langle Y \rangle^{(i)})_1$$

$$\langle X \rangle^{(i)} \rightarrow (X)_i, \quad \langle Y \rangle^{(i)} \rightarrow (Y)_i \quad \text{and} \quad \langle Z \rangle^{(i)} \rightarrow (Z)_i$$

Now we prove the equivalence of $G$ and $G'$. To prove that $L(G) \subseteq L(G')$, we notice that case (1) is similar to the classical proof of the Chomsky normal form

and is not pursued. Similarly, for case (2) the right part $XYZ$ is first encoded in the first segment, then moved one by one into the proper position.

We omit the proof that $L(G) \supseteq L(G')$, which is straightforward.

□

*Note 13.* We can assume that the non-terminal alphabets of each list component are disjoint, that is $V_N = \biguplus_{i=1}^{n} V_N^{(i)}$, the disjoint union of $n$ alphabets $V_N^{(i)} = \{A^{(i)}\}$. Accordingly, in each production the $i$-th component is a string in $\left(V_N^{(i)}\right)^*$, for every $1 \leq i \leq n$.

*Note 14.* If we suppose $G = (V_N, V_T, P, S^{(1)})$, for $n \geq 2$, with $V_N = \biguplus_{i=1}^{n} V_N^{(i)}$ (see the previous note), we can construct an equivalent normal form for $G$, called *strong normal form*, the productions of which are as follows:

$A^{(1)} \rightarrow (\alpha)_1$ where $\alpha \in \left(V_N^{(1)}\right)^*$ and $|\alpha| = 2$

$A^{(1)} \rightarrow (A^{(i)})_i$ where $A^{(i)} \in V_N^{(i)}$ and $1 < i \leq n$

$A^{(i)} \rightarrow (A^{(1)})_1$ where $A^{(i)} \in {V_N}^{(i)}$ and $1 < i \leq n$

$A^{(1)} \rightarrow a$ where $a \in V_T$

$S^{(1)} \rightarrow \varepsilon$ if and only if $\varepsilon \in L(G)$

Notice that a $D^n$ grammar with $n = 1$, i.e. a cf grammar, in strong normal form reduces to one in Chomsky normal form (because the $2^{nd}$ and $3^{rd}$ production types above do not apply); therefore the strong normal form is a generalization to $D^n$ grammars of the Chomsky normal form of cf grammars.

*Note 15.* We could analogously construct a (strong) normal form for $n$-pd automata, proving that for any $n$-pd automaton $M$, with $n \geq 2$, there exists an equivalent machine $M' = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, such that:

$\Gamma = \biguplus_{i=1}^{n} \Gamma^{(i)}$, where the $\Gamma^{(i)}$'s are disjoint memory alphabets

$\delta(q, \varepsilon, A^{(1)}) \subseteq \Delta_1 \cup \Delta_2$, where

$\Delta_1 = \{(q', B^{(1)}C^{(1)}, \varepsilon, \varepsilon, \ldots, \varepsilon) | \quad q' \in Q\}$
$\Delta_2 = \{(q', \varepsilon, \ldots, \varepsilon, A^{(i)}, \varepsilon, \ldots, \varepsilon) | \quad q' \in Q \text{ and } B^{(i)} \in \Gamma^{(i)}, \text{ for } 1 < i \leq n\}$

for any $1 < i \leq n$ if $(q_0, \varepsilon, \ldots, \varepsilon) \in \delta(q_0, \varepsilon, Z_0)$

$\delta(q, \varepsilon, A^{(i)}) \subseteq \{(q', B^{(1)}, \varepsilon, \ldots, \varepsilon) | \quad q' \in Q\}$

$\delta(q, a, A^{(1)}) \subseteq \{(q', \varepsilon, \ldots, \varepsilon) | \quad q' \in Q\}$

which is equivalent to $M$ and recognizes by empty store.

Equivalence of $PD^n$ automata and $D^n$ grammars As described in the introduction, a pd automaton performs the depth-first left-to-right parse of the strings generated by a cf (i.e. $D$) grammar in Greibach normal form. This correspondence can be extended to $PD^n$ machines and to $D^n$ grammars, as stated by the next result.

**Lemma 16.** *A language $L$ is recognized by a $n$-pd automaton $M$ by empty tape if and only if $L \in \mathcal{L}_{D^n}$, for some $n \geq 1$, i.e. $\mathcal{L}_{PD^n} = \mathcal{L}_{D^n}$.* $\square$

**Proof** Let $L$ be a language recognized by empty store by a $n$-pd automaton, in (strong) normal form, $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$. We shall construct a $D^n$ grammar $G = (V_N, V_T, P, S)$ with the input alphabet of $M$ as terminal alphabet, i.e. $V_T = \Sigma$, and a non-terminal alphabet $V_N = \{\Gamma \times Q^{2n}\} \uplus \{\Gamma \times Q^2\}$, in order to encode the states of the automaton; the axiom $S$ is the initial memory symbol of $M$, i.e. $S = Z_0$. The productions in $P$ have the form:

1. $\forall q \in Q$ and iff $\varepsilon \in L$

$$Z_0 \rightarrow (\langle Z_0; q_0, q; q, q; \ldots; q, q\rangle)_1 \quad \text{and} \quad Z_0 \rightarrow \varepsilon$$

2. $\forall q'_{2h} \in Q$, with $2 \leq h \leq n$, and iff $(q'_1, B^{(1)}C^{(1)}, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(1)})$

$$\langle A^{(1)}; q_1, q_2; q_3, q_4; \ldots; q_{2n-1}, q_{2n}\rangle \rightarrow \\ (\langle B^{(1)}; q'_1, q'_2; q_3, q'_4; \ldots; q_{2n-1}, q'_{2n}\rangle \langle C^{(1)}; q'_2, q_2; q'_4, q_4; \ldots; q'_{2n}, q_{2n}\rangle)_1$$

3. iff $(q_2, \varepsilon, \ldots, B^{(j)}, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(1)})$

$$\langle A^{(1)}; q_1, q_2; q_3, q_3; \ldots; q_{2j-3}, q_{2j-3}; q_{2j-1}, q_{2j}; q_{2j+1}, q_{2j+1}; \ldots; q_{2n-1}, q_{2n-1}\rangle \\ \rightarrow (\langle B^{(j)}; q_{2j}, q_{2j-1}\rangle)_j$$

4. $\forall q\ q_{2h+1} \in Q$, with $1 \leq h \leq j-1$, and iff $(q'_1, C^{(1)}, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(j)})$

$$\langle A^{(j)}; q_1, q_2\rangle \rightarrow (\langle C^{(1)}; q'_1, q; q_3, q; \ldots; q_{2j-3}, q; q_2, q; q_1, q; \ldots; q_1, q\rangle)_1$$

5. iff $(q_2, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, a, A^{(1)})$

$$\langle A^{(1)}; q_1, q_2; q_3, q_3; \ldots; q_{2n-1}, q_{2n-1}\rangle \rightarrow a$$

The grammar $G$ simulates the behaviour of the automaton $M$ by guessing (nondeterministically) the correct sequence of states, coded in the first and $(2h+2)$-th state component of the first pd tape symbol ($h > 1$). This fact will be completely proved by induction in Appendix A.

Conversely, given $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$, from Statement 10 we can assume that $G$ contains productions of the types

$$A \rightarrow (\alpha_1)_1(\alpha_2)_2 \ldots (\alpha_n)_n, \quad \text{with } \alpha_1 \in V_N^*, \quad \text{and} \quad A \rightarrow a, \quad \text{with } a \in V_T$$

We construct the one-state $n$-pd automaton $M = (q_0, \Sigma, \Gamma, \delta, q_0, Z_0)$ as specified (the state is omitted):

$$\Sigma = V_T \qquad \Gamma = V_N \qquad Z_0 = S$$

$$(\alpha_1, \ldots, \alpha_n) \in \delta(\varepsilon, A) \qquad \text{if and only if } A \rightarrow (\alpha_1)_1 \ldots (\alpha_n)_n \in P$$

$$(\varepsilon, \ldots, \varepsilon) \in \delta(a, A) \qquad \text{if and only if } A \rightarrow a \in P$$

where $A \in V_N$, $a \in V_T$ and $\alpha_j \in V_N^*$, for any $1 \leq j \leq n$. The automaton $M$ recognizes by empty store. The proof that $L(M) = L(G)$ is straightforward and is omitted.

□

The previous results are summarized by the next statement.

**Theorem 17.** *For any mpd language $L$ the following are equivalent:*

- *$L$ is recognized by a $n$-pd automaton, recognizing by empty tape;*
- *$L$ is recognized by a $n$-pd automaton, recognizing by final state;*
- *$L$ is generated by a $D^n$ grammar.*

*for some $n \geq 1$.* □

This theorem establishes a close parallelism between cf and mpd languages.

# 3 Properties

## 3.1 Properties of $n$-pd languages

We prove several properties holding for any member of the superfamily of mpd or multi-depth languages. Take a language $L = L(G)$, with $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$. The productions of the normal form grammar generating $L - \{\varepsilon\}$ are length increasing, hence

**Statement 18.** Any language $L \in \mathcal{L}_{D^n}$ is context-sensitive, for any $n \geq 1$. □

**Definition 19.** For any mpd grammar $G \in \mathcal{G}_{D^n}$, let $G_{CF} = (V_N, V_T, P_{CF}, S)$ be the cf grammar with

$$P_{CF} = \{A \rightarrow w\alpha_1\alpha_2 \ldots \alpha_n | \quad A \rightarrow w(\alpha_1)_1(\alpha_2)_2 \ldots (\alpha_n)_n \in P\}$$

This grammar is called the underlying cf grammar of $G$. □

The existence of the underlying cf grammar is significant. For every string $w \in L(G)$ there exists a string $w' \in L(G_{CF})$ that is a permutation of $w$, and conversely. Since the Parikh image of $L(G_{CF})$ is semilinear [20], the Parikh image of $L = L(G)$ is semilinear, too, and we have proved the following

**Statement 20.** The Parikh image of every depth-$n$ language is semilinear, for any $n \geq 1$. □

Any grammar $G \in \mathcal{G}_{D^n}$ can be put in reduced form, i.e. all non-generating or looping derivations can be excluded. In fact, consider that a $D^n$ grammar $G$ produces non-generating or looping derivations if and only if the underlying cf grammar $\mathcal{G}_{CF}$ does, too. Hence to reduce $G$ it suffices to process $\mathcal{G}_{CF}$ reducing it instead (this only requires stripping off some productions), and then to go back to $G$. Moreover, the depth-$n$ language $L = L(G)$ is empty if and only if the cf language $L_{CF} = L(G_{CF})$ is empty. Therefore

**Statement 21.** The emptiness problem of any depth-$n$ language $L \in \mathcal{L}_{D^n}$ is decidable, for any $n \geq 1$.

Finally, note that a depth-$n$ language $L$ contains the empty string $\varepsilon$ if and only if $L_{CF}$ does, too. Hence

**Statement 22.** The problem of deciding whether a depth-$n$ language $L \in \mathcal{L}_{D^n}$ contains the empty string $\varepsilon$ is decidable, for any $n \geq 1$.

The following statements can be proved essentially by means of the same proofs used for cf grammars, and hold for any $n \geq 1$.

**Statement 23.** $\mathcal{L}_{D^n}$ is closed with respect to union. □

**Statement 24.** $\mathcal{L}_{D^n}$ is closed with respect to (erasing) alphabetic homomorphism. □

**Statement 25.** $\mathcal{L}_{D^n}$ is closed with respect to inverse non-erasing alphabetic homomorphism. □

Since in the sequel we shall prove that every family $\mathcal{L}_{D^n}$ is an AFL, Statements 24 and 25 hold even if the hom.'s are arbitrary and erasing.

**Statement 26.** $\mathcal{L}_{D^n}$ is closed with respect to catenation and Kleene star. □

**Proof** We distinguish the two operations, catenation and Kleene star.
*Catenation.* Let $L_1 = L(G_1)$ and $L_2 = L(G_2)$, where $V_N^{(1)} \cap V_N^{(2)} = \emptyset$. Construct $G = (V_N, V_T, P, S)$ as follows.

$$V_N = V_N^{(1)} \cup V_N^{(2)} \cup \{S\} \quad \text{where } S \text{ is a new non} - \text{terminal}$$

$$P = P_1 \cup P_2 \cup \begin{cases} S \rightarrow (S_1)_1 (S_2)_n \\ S_2 \rightarrow (S_2)_1 \end{cases}$$

Clearly it holds $L(G) = L_1 L_2$.
*Kleene star.* Let $L_1 = L(G_1)$. Construct $G = (V_N, V_T, P, S)$ as follows.

$$V_N = V_N^{(1)} \cup \{S, S_1'\} \quad \text{and} \quad P = P_1 \cup \begin{cases} S \rightarrow (S_1)_1 (S_1')_n \\ (S_1') \rightarrow (S_1)_1 (S_1')_n \\ (S_1') \rightarrow \varepsilon \\ S \rightarrow \varepsilon \end{cases}$$

Clearly it holds $L(G) = L_1^*$.

□

**Statement 27.** $\mathcal{L}_{D^n}$ is closed with respect to the intersection with regular languages. □

**Proof** Let $L \in \mathcal{L}_{D^n}$. Given a finite automaton recognizing the regular language $R$ and the $n$-pd recognizer by final states of $L$, construct the Cartesian product machine.

□

Szilard language The definition of *leftmost Szilard language* for a cf grammar [11] can be extended to $D^n$ grammars.

**Definition 28.** Let $G \in \mathcal{G}_{D^n}$ and let $E$ be the set of the unique labels of its productions, then the leftmost Szilard language of $G$ is $Z_L(G) = \{y \in E^+ | \quad D$ is a derivation $S \stackrel{*}{\Rightarrow}_G x$ and $y$ is the sequence of productions applied in $D\}$. The string $y$ is called the *control word* of the derivation $D$ of the string $x$. □

**Statement 29.** $Z_L(G) \in \mathcal{L}_{D^n}$, for every grammar $G \in \mathcal{G}_{D^n}$. □

**Proof** Let $G = (V_N, V_T, P, S)$ be a $D^n$ grammar. Consider the grammar $G' = (V_N, E, P, S)$, where

$$P' = \{X \to e(\alpha_1)_1(\alpha_2)_2 \ldots(\alpha_n)_n | \quad e : X \to w(\alpha_1)_1(\alpha_2)_2 \ldots(a_n)_n \in P\}$$

so clearly $L(G) = Z_L(G)$.

□

Considering now the Szilard language $Z_L(G_{CF})$ of the underlying cf grammar, we have immediately:

**Statement 30.** For every $G \in \mathcal{G}_{D^n}$, it holds $Z_L(G) \subseteq Z_L(G_{CF})$. □

These statements will be used in the proof of the next result.
Generalized Dyck language As cf (i.e. $\mathcal{L}_D$) languages as their generator have the Dyck language, which characterizes the family, for each $\mathcal{L}_{D^n}$ there exists a corresponding generator, a generalized Dyck language to be next defined.

The alphabet of the Dyck language consists of finitely many pairs $a, a^{(1)}$; $b, b^{(1)}$; ... of symbols. The alphabet of the generalized Dyck language consists of finitely many $(n+1)$-tuples of symbols $aa^{(1)} \ldots a^{(n)}$; $b, b^{(1)} \ldots b^{(n)}$; ....

**Definition 31.** The *generalized Dyck language* over an alphabet $\Sigma$, corresponding to $n \geq 1$ pd tapes, shortly $D(\Sigma, n)$, is defined as follows.

Let $\Sigma^{(i)}$, with $1 \leq i \leq n$, be marked copies of $\Sigma$, and let $\tilde{\Sigma} = \biguplus_{i=1}^{n} \Sigma^{(i)} \uplus \Sigma$. Then $D(\Sigma, n)$ is defined by the following $D^n$ grammar $G = (V_N, \tilde{\Sigma}, P, S)$

$$V_N = \{S\} \cup \{A^{(i)} | \quad \forall a \in \Sigma \quad \forall i \; 1 \leq i \leq n \quad A = \text{ upper case copy of } a\}$$

$$P = \begin{cases} S \to \varepsilon \\ S \to a(SA^{(1)})_1(A^{(2)})_2 \ldots(A^{(n)})_n \\ A^{(i)} \to a^{(i)}(S)_1 \end{cases}$$

for any $a \in \Sigma$ and for any $1 \leq i \leq n$. □

As an example we give the $D^2$ grammar $G = (V_N, \tilde{\Sigma}, P, S)$ generating the generalized Dyck language $D(\{a, b\}, 2)$:

$$V_N = \{S, A^{(1)}, A^{(2)}, B^{(1)}, B^{(2)}\} \qquad \tilde{\Sigma} = \{a, a^{(1)}, a^{(2)}, b, b^{(1)}, b^{(2)}\}$$

$$P = \begin{cases} S \to \varepsilon \\ S \to a(SA^{(1)})_1(A^{(2)})_2 \\ S \to b(SB^{(1)})_1(B^{(2)})_2 \\ A^{(1)} \to a^{(1)}(S)_1 \\ A^{(2)} \to a^{(2)}(S)_1 \\ B^{(1)} \to b^{(1)}(S)_1 \\ B^{(2)} \to b^{(2)}(S)_1 \end{cases}$$

Here follows an example of a derivation in $G$.

$$\begin{aligned} S &\Rightarrow a(SA^{(1)})_1(A^{(2)})_2 \\ &\Rightarrow ab(SB^{(1)}A^{(1)})_1(B^{(2)}A^{(2)})_2 \\ &\Rightarrow ab(B^{(1)}A^{(1)})_1(B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}(SA^{(1)})_1(B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}a(SA^{(1)}A^{(1)})_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}a(A^{(1)}A^{(1)})_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}(SA^{(1)})_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}(A^{(1)})_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}a^{(1)}(S)_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}a^{(1)}(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}a^{(1)}a^{(2)}(S)_1(B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}a^{(1)}a^{(2)}a(SA^{(1)})_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow abb^{(1)}aa^{(1)}a^{(1)}a^{(2)}a(A^{(1)})_1(A^{(2)}B^{(2)}A^{(2)})_2 \\ &\Rightarrow \ldots \Rightarrow abb^{(1)}aa^{(1)}a^{(1)}a^{(2)}aa^{(1)}a^{(2)}b^{(2)}a^{(2)} = x \end{aligned}$$

Examining the string $x$ it is immediate to notice that its projections over $\Sigma \cup \Sigma^{(1)}$ and over $\Sigma \cup \Sigma^{(2)}$ are Dyck strings. Moreover, the projection over $\Sigma \cup \Sigma^{(1)}$ of each prefix (e.g. $abb^{(1)}aa^{(1)}a^{(1)}$) of $x$, immediately followed by a character in $\Sigma^{(2)}$, is in the Dyck language over $\Sigma \cup \Sigma^{(1)}$. Figure 3 shows these projections for the generalized Dyck string of the above derivation. Next we present a definition in terms of cancellation rules. Let $\Sigma_{(1)}$ and $\Sigma_{(2)}$ be two indexed copies of $\Sigma$, and consider the homomorphism $h : \left(\tilde{\Sigma}\right)^* \to \left(\Sigma_{(1)} \cup \Sigma_{(2)} \cup \Sigma^{(1)} \cup \Sigma^{(2)}\right)^*$ defined by

$$\begin{aligned} h(a) &= a_{(2)}a_{(1)} && \text{for every } a \in \Sigma \\ h(a^{(1)}) &= a^{(1)} && \text{for every } a^{(1)} \in \Sigma^{(1)} \\ h(a^{(2)}) &= a^{(2)} && \text{for every } a^{(2)} \in \Sigma^{(2)} \end{aligned}$$

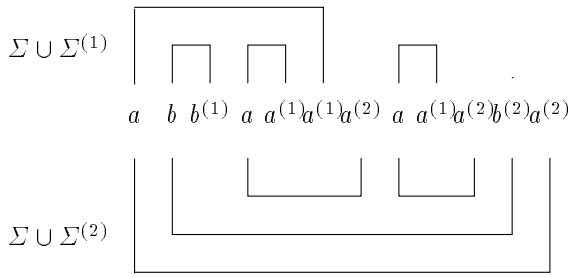


**Fig. 3.** *Projections of the generalized Dyck language.*

and apply the following rewriting rule

$$ux_{(1)}vx^{(1)}w = uvw$$

to the homomorphic image of $x$ through $h$, if and only if $v \in \Sigma^*_{(2)}$ and $u, w \in \left(\Sigma_{(1)} \cup \Sigma_{(2)} \cup \Sigma^{(1)} \cup \Sigma^{(2)}\right)^*$. For instance, suppose $x$ is the string of the above derivation, then $h(x)$ becomes

$$a_{(2)}a_{(1)}b_{(2)}b_{(1)}b^{(1)}a_{(2)}a_{(1)}a^{(1)}a^{(1)}a^{(2)}a_{(2)}a_{(1)}a^{(1)}a^{(2)}b^{(2)}a^{(2)}$$

and can be rewritten as follows

$$a_{(2)}a_{(1)}b_{(2)}b_{(1)}b^{(1)}a_{(2)}a^{(1)}a^{(2)}a_{(2)}a_{(1)}a^{(1)}a^{(2)}b^{(2)}a^{(2)}$$

$$a_{(2)}a_{(1)}b_{(2)}a_{(2)}a^{(1)}a^{(2)}a_{(2)}a_{(1)}a^{(1)}a^{(2)}b^{(2)}a^{(2)}$$

$$a_{(2)}b_{(2)}a_{(2)}a^{(2)}a_{(2)}a_{(1)}a^{(1)}a^{(2)}b^{(2)}a^{(2)}$$

$$a_{(2)}b_{(2)}a_{(2)}a^{(2)}a_{(2)}a^{(2)}b^{(2)}a^{(2)}$$

which is a string of the Dyck language over $\Sigma_{(2)} \cup \Sigma^{(2)}$ and can be reduced to the empty word by means of the usual cancellation rules for the Dyck language.

Notice that the introduction of the homomorphism $h$ and of the new alphabets $\Sigma_{(1)}$ and $\Sigma_{(2)}$ has two purposes: to mark the positions of the characters belonging to $\Sigma$ in the original string after the application of the cancellation rule and to control their correct matching with the characters in $\Sigma^{(2)}$. This example leads to the next rules.

*Note 32.* It is possible to define the generalized Dyck language by means of a generalized cancellation rule, as follows. We introduce the new alphabets $\Sigma_{(i)}$, with $i = 1, 2, \ldots, n$, which are disjoint indexed copies of $\Sigma$, and the homomorphism

$$h : (\tilde{\Sigma})^* \rightarrow \left(\Sigma_{(1)} \cup \Sigma_{(2)} \cup \ldots \cup \Sigma_{(n)} \cup \Sigma^{(1)} \cup \Sigma^{(2)} \cup \ldots \cup \Sigma^{(n)}\right)^*$$

defined by

$$h(a) = a_{(n)} \dots a_{(2)} a_{(1)} \quad \text{and} \quad h(a^{(i)}) = a^{(i)}$$

for any $a \in \Sigma$, $a^{(i)} \in \Sigma^{(i)}$ and $i = 1, 2, \dots, n$. Then

$$D(\Sigma, n) = \{ w \in \tilde{\Sigma}^* | \quad h(w) \cong \varepsilon \}$$

where $\cong$ is the reflexive and transitive closure of the transformation

$$u a_{(i)} v a^{(i)} w = uvw$$

with $v \in \Sigma^*_{(i+1)} \cup \dots \cup \Sigma^*_{(n)}$ and $u, w \in (h(\tilde{\Sigma}))^*$. If $n = 1$ it is easily proved that the reflexive and transitive closure of the rewriting rule

$$u a a' w = uw$$

where $u, w \in (\Sigma \cup \Sigma')^*$, coincides with the congruence generated by $aa' = \varepsilon$, which is the usual cancellation rule for the Dyck language [2].

The proof that the above definition of generalized Dyck language is equivalent to Definition 31 can be obtained by induction on the number of characters belonging to $\Sigma$ occurring in a string.

Chomsky-Schützenberger property We extend the classical homomorphic characterization theorem by N. Chomsky and M. P. Schützenberger from cf [11, 14, 20] to mpd languages.

**Theorem 33.** *A language $L$ belongs to $\mathcal{L}_{D^n}$ if and only if there exist an alphabet $\Sigma$, a homomorphism $\Phi$ and a regular language $R_2$ such that $L = \Phi(D(\Sigma, n) \cap R_2)$, where $D(\Sigma, n)$ is the generalized Dyck language relative to $n$ pd tapes, for some $n \geq 1$.* □

**Proof** Let $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$ be in strong normal form (see Note 14) and let $\Sigma$ be the set of the labels of its productions. The language $D(\Sigma, n)$ is as in Definition 31. Let $p$ denote the projection of $\tilde{\Sigma}$ over $\Sigma$, i.e. the homomorphism defined by $p(e) = e$, for every $e \in \Sigma$, and by $p(e^{(i)}) = \varepsilon$, for every $e^{(i)} \in \Sigma^{(i)}$ with $1 \leq i \leq n$. Then clearly $p(D(\Sigma, n)) = \Sigma^*$, since the generalized Dyck language imposes matches between $e \in \Sigma$ and the corresponding $e^{(i)} \in \Sigma^{(i)}$, but allows any sequence of $e$'s.

We construct a regular language $R_2$ such that $p(D(\Sigma, n) \cap R_2)$ is $Z_L(G)$, the leftmost Szilard language of $G$. We also define the homomorphism $\Phi : \tilde{\Sigma} \to V_T^*$ by

$$\Phi(e) = w \in V_T \qquad \text{if } e : X \to w \in P, \text{ otherwise } \Phi(e) = \varepsilon$$

$$\Phi(e^{(i)}) = \varepsilon \qquad \text{for any } 1 \leq i \leq n$$

in order to obtain from each Szilard word the corresponding word in $L(G)$.

We shall show that given a derivation $u = e_{i_1} \dots e_{i_n} : S \overset{*}{\Rightarrow}_G x \in L(G)$, a unique $\hat{u} \in D(\Sigma, r) \cap R_2$ can be built such that $\Phi(\hat{u}) = \Phi(u) = x$ and $p(\hat{u}) = u$.

To construct such a $\hat{u}$ we introduce a new grammar $G'' = (V''_N, \tilde{\Sigma}, P'', S) \in \mathcal{G}_{D^n}$. The grammar $G''$ is built as follows. Let

$$V''_N = V_N \cup V_1 \cup \ldots \cup V_n \quad \text{where } V_j = \{E_i^{(j)} | \quad e_i^{(j)} \in \Sigma^{(j)}\} \quad \text{for } 1 \leq j \leq n$$

$$P'' = \begin{cases} E_i^{(j)} \to e_i^{(j)} & \text{if } E_i^{(j)} \in V_j \\ X \to e_i \left(E_i^{(1)}\right)_1 \ldots \left(E_i^{(n)}\right)_n & \text{if } e_i : X \to w \in P \\ X \to e_i \left(E_i^{(1)}\right)_1 \ldots \left(\alpha_j E_i^{(j)}\right)_j \ldots \left(E_i^{(n)}\right)_n & \text{if } e_i : X \to (\alpha_j)_j \in P \end{cases}$$

for every $j = 1, 2, \ldots, n$. Then we define the regular (locally testable) language $R_2$ by specifying the initial characters and the adjacent pairs to be allowed.

The character $e_i$ is the first character of a string $\hat{u} \in R_2$ if and only if it is a label of a production $S^{(1)} \to \ldots \in P$. Any other character in $\tilde{\Sigma}$ is a forbidden initial character. The adjacent pairs are all the pairs $e_i^{(j)} e_k^{(h)}$ occurring as substrings of $\hat{u} \in L(G'')$. Define the regular language $R_{e_i{}^{(j)} e_k{}^{(h)}} = \tilde{\Sigma}^* e_i{}^{(j)} e_k{}^{(h)} \tilde{\Sigma}^*$. The pairs that do not occur as substrings of $\hat{u}$ are called forbidden pairs. A pair $e_i^{(j)} e_k^{(h)}$ is a forbidden pair if and only if the intersection between $L(G'')$ and the regular language $R_{e_i{}^{(j)} e_k{}^{(h)}}$ is empty. Since $\mathcal{L}_{D^n}$ is closed with respect to the intersection with regular languages, $L(G'') \cap R_{e_i{}^{(j)} e_k{}^{(h)}}$ and $L(G'') \cap e_i \tilde{\Sigma}^*$ are in $\mathcal{L}_{D^n}$. But for a $D^n$ grammar $G$ the emptiness problem of $L(G)$ is decidable, as proved in Statement 21. Consider the underlying cf grammar $G_{CF}$, then $L(G)$ is empty if and only if $L(G_{CF})$ is empty, and the emptiness of a cf language is decidable. Hence the membership problem for the set of forbidden initial characters and for the set of forbidden pairs is decidable. Let $R_2$ be the complement of the union of the $R_{e_i{}^{(j)} e_k{}^{(h)}}$'s, for all the forbidden pairs $e_i^{(j)} e_k^{(h)}$, and of the $e_i \tilde{\Sigma}^*$'s, where $e_i$ is any forbidden initial character. All the strings in $R_2$ start with an admissible initial character and contain only admissible pairs.

Therefore $L(G'')$ is contained in $R_2$ by definition of $R_2$. From Definition 31 it follows immediately that $L(G'')$ is contained in $D(\Sigma, n)$. Moreover, for every $\hat{u} \in L(G'')$ the string $p(\hat{u})$ is the control word of a derivation $S^{(1)} \stackrel{*}{\Rightarrow}_G x$, where $x \in L(G)$, and, conversely, given a derivation $u = e_{i_1} \ldots e_{i_n} : S^{(1)} \stackrel{*}{\Rightarrow}_G x \in L(G)$, there exists a string $\hat{u} \in L(G'')$ such that $p(\hat{u}) = u$, by the definition of $G''$. Finally $\Phi(u) = \Phi(p(\hat{u})) = x$, hence $L(G) = \Phi(L(G'')) \subseteq \Phi(D(\Sigma, n) \cap R_2)$. The opposite inclusion $D(\Sigma, n) \cap R_2 \subseteq L(G'')$ is straightforward.

The converse follows immediately from Statements 24 and 27.

□

Pumping lemma A periodicity property is proved which generalizes the so-called "pumping lemma" of cf languages (Bar-Hillel or Ogden lemma) [20]. This result will be used to prove that certain languages do not belong to the $n$-pd family, for any $n \geq 1$. First we need the following definitions.

**Definition 34.** (*merge*) Let $\overline{x} = (x_1)(x_2)\ldots(x_r)$ and $\overline{y} = (y_1)(y_2)\ldots(y_s)$, for $x_i, y_j \in \Sigma^*$, "(", ")" $\notin \Sigma^*$ and $r, s \geq 1$, be two lists (see Definition 4) over the

alphabet $\Sigma$. The *marked merge* of $\overline{x}$ and $\overline{y}$, shortly $\overline{m}(\overline{x},\overline{y})$, is a list defined in the following way:

$$\overline{m}(\overline{x},\overline{y}) = (x_1y_1)\dots(x_ry_r)(y_{r+1})\dots(y_s) \quad \text{if } r \le s \text{ (and similarly if } r > s)$$

Moreover, $m(\overline{x},\overline{y})$ indicates the string $x_1y_1x_2y_2\dots x_ry_ry_{r+1}\dots y_s$ of $V_T^*$. The concatenation operation is defined on lists by

$$\overline{xy} = (x_1)\dots(x_ry_1)\dots(y_s)$$

The concatenation of a list $\overline{x}$ with a string $v$ is

$$\overline{x}v = (x_1)\dots(x_rv) \quad \text{and} \quad v\overline{x} = (vx_1)\dots(x_r)$$

□

*Remark.* Since merge is associative, it is possible to define

$$\overline{m}(\overline{x},\overline{y},\overline{z}) = \overline{m}(\overline{x},\overline{m}(\overline{y},\overline{z})) = \overline{m}(\overline{m}(\overline{x},\overline{y}),\overline{z})$$

**Notation.** For each string $x \in \Sigma^*$, the string $x^R$ is the mirrored image of $x$.

**Theorem 35.** *(Pumping Lemma) Let $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$, for some $n \ge 1$, and $L = L(G)$. Then there exist two integers $p, q > 0$ such that:*

$$\forall z \in L, \text{ with } |z| > p, \text{ it holds } z = z_1z_2\dots z_h, \text{ for some } h \text{ with } 1 \le h \le n \quad (4)$$

*where*

$$\begin{aligned}
z_1 &= z_{11}\\
z_2 &= (z_{21})^R\\
z_3 &= z_{32}(z_{31})^R\\
&\dots\dots\dots\dots\dots\\
z_i &= z_{i,2^{i-2}}(z_{i,2^{i-2}-1})^R z_{i,2^{i-2}-2}\dots(z_{i1})^R, \text{ for any } 3 \le i \le h\\
z_{ij} &= m(\overline{x}_{ij}, \overline{y}_{ij} v_{ij} w_{ij} \overline{u}_{ij})
\end{aligned}$$

*with*

$$\overline{x}_{ij} = (x_{ij1})(x_{ij2})\dots(x_{ij2^{n-h}+1}) \quad \text{and} \quad \overline{y}_{ij} = (y_{ij1})(y_{ij2})\dots(y_{ijr})$$

*where $y_{ijr} = \varepsilon$ and $2^{n-h-1} \le r-1 \le 2^{n-h}-1$, if $n-h-1 \ge 0$, otherwise $r = 1$, and with*

$$\overline{u}_{ij} = (u_{ijr})(u_{ijr+1})\dots(u_{ijt})$$

*where $t = 2^{n-h}$, and also with*

$$yvu \ne \varepsilon \quad \text{and} \quad |w| < q \quad (5)$$

*where $y, v, u$ and $w$ are the catenations of the strings $y_{ij}, v_{ij}, u_{ij}$ and $w_{ij}$ ($y_{ij}$ and $u_{ij}$ are the unmarked copies of $\overline{y}_{ij}$ and $\overline{u}_{ij}$), respectively.*

*The existence of the above decomposition of the string $z$ implies that the following string:*

$$z' = z'_1 z'_2 \dots z'_h \quad \textit{is in } L \tag{6}$$

*where*

$$\begin{aligned}
z'_1 &= z'_{11} \\
z'_2 &= (z'_{21})^R \\
z'_3 &= z'_{32}(z'_{31})^R \\
&\dots\dots\dots\dots\dots \\
z'_i &= z'_{i,2^{i-2}}(z'_{i,2^{i-2}-1})^R z'_{i,2^{i-2}-2} \dots (z'_{i1})^R, \textit{ for any } 3 \le i \le h \\
z'_{ij} &= m(\overline{x}_{ij}, \overline{y}_{ij} v_{ij} y_{ij} v_{ij} w_{ij} u_{ij} \overline{u}_{ij})
\end{aligned}$$

*Notice that in the strings $z'_{ij}$ both the marked and the unmarked forms of the lists $\overline{y}_{ij}$ and $\overline{u}_{ij}$ occur.* □

*Note 36.* If $G \in \mathcal{G}_D$ this lemma reduces to the well-known "pumping lemma" for cf languages. In fact, in this case $h = 1$, hence the statement claims that there exist positive non-null integers $p$ and $q$ such that if $z \in L(G)$ and $|z| > p$, then

$$z = z_1 = z_{11} = m((x_{111})(x_{112}), (\varepsilon v_{11} w_{11} u_{111})) = x_{111} v_{11} w_{11} u_{111} x_{112}$$

for some $x_{111}, v_{11}, w_{11}, u_{111}, x_{112} \in {V_T}^*$, with $v_{11} u_{111} \neq \varepsilon$ and $|w_{11}| < q$, and

$$z' = m((x_{111})(x_{112}), (v_{11} v_{11} w_{11} u_{111} u_{111})) = x_{111} v_{11}^2 w_{11} u_{111}^2 x_{112} \in L$$

In order to clarify the situation we state completely the "pumping lemma" also for $n = 2$ and we give an idea of the proof in Figures 4, 5, 6 and 7. Let $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^2}$ and $L = L(G)$. Then there exist integers $p, q > 0$ such that for every $z \in L$, with $|z| > p$, it is either

$$z = x_{111} y_{111} x_{112} v_{11} w_{11} u_{112} x_{113}$$

with $y_{111} v_{11} u_{112} \neq \varepsilon$ and $|w_{11}| < q$ (case $h = 1$), or

$$z = x_{111} v_{11} w_{11} u_{111} x_{112} x_{212} u_{212} w_{21} v_{21} x_{211}$$

with $v_{11} v_{21} u_{111} u_{112} \neq \varepsilon$ and $|w_{11} w_{21}| < q$ (case $h = 2$), and

$$z' = x_{111} y_{111} x_{112} v_{11} y_{111} v_{11} w_{11} u_{112}^2 x_{113}$$

or

$$z' = x_{111} v_{11}^2 w_{11} u_{111}^2 x_{112} x_{212} u_{212}^2 w_{21} v_{21}^2 x_{211}$$

respectively, are in $L$.

**Fig. 5.** *Proof of the "pumping lemma": $h = 1$, case $(b)$.*

**Fig. 7.** *Proof of the "pumping lemma": $h = 2$, case $(b)$.*

*Hint of the proof.* Let $L = L(G)$, where $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$ is in strong normal form (see Note 14). Let $G_{CF}$ be the underlying cf grammar. Then $L' = L(G_{CF})$ is a cf language the elements of which are permutations of words in $L$. For a string $w \in L$ and a derivation $S \stackrel{*}{\Rightarrow}_G w$, consider the corresponding derivation in $G_{CF}$, $S \stackrel{*}{\Rightarrow}_{G_{CF}} \tilde{w}$, so that $w$ is a permutation of $\tilde{w}$. Let $T$ be the syntax tree of $\tilde{w}$ in $G_{CF}$. By a suitable order of visit of $T$ it is possible to obtain $w$. This ordering is stored in $T$ by assigning an apex to each internal node. Thus $T$ can be considered the syntax tree of $w$, too.

If $|w| > h$, there exists in $T$ at least one path with $1 + \lceil \log_2 h \rceil$ nodes. Let now $m = |V_N|$. If we consider a string $z$ in $L$ with $|z| > 2^{m+1}$, the syntax tree of $z$ contains at least one path with $m+2$ nodes. Thus in this path there exists a non-terminal $A$ which occurs twice; we can suppose - from the form of the productions in $G$ - that this non-terminal belongs to $V_N^{(1)}$. Denote by $A_1^{(1)}$ its occurrence closest to the root of the tree and by $A_2^{(1)}$ the following one. We can choose $A_1^{(1)}$ and $A_2^{(1)}$ so that in the path $[A_1^{(1)}, A_2^{(1)}]$ at least one bifurcation occurs. Otherwise we could delete all the paths of the kind $[A_1^{(1)}, A_2^{(1)}]$, obtaining a syntax tree of a permutation $z''$ of $z$, with $z'' \in L$, containing no path of length exceeding $m+2$. Furthermore, we can choose the path $[A_1^{(1)}, A_2^{(1)}]$ in such a way that it is the most distant from the root, among the paths satisfying the previous conditions.

From now on the proof is analogous to the cf case. When the non-terminal $A_2^{(1)}$ occurs as the first symbol of the sequence to be rewritten, it can be expanded with the same derivation used in the expansion of $A_1^{(1)}$, obtaining a new string $z'$ in which the substrings of $z$ generated by non-terminals in $[A_1^{(1)}, A_2^{(1)}]$ occur twice. The positions where such substrings occur depend on the names of the apices of the non-terminals belonging to $[A_1^{(1)}, A_2^{(1)}]$, i.e. depend on the list components, or pd tapes, where the sons of $A_1^{(1)}$ have been rewritten. So we must distinguish several cases depending on the names of the apices of the non-terminals occurring in the interval $[A_1^{(1)}, A_2^{(1)}]$. The lengthy complete proof for the case $n = 3$ is given in Appendix B. □

In order to apply the "pumping lemma" we need the following definition.

**Definition 37.** A language $L$ over an alphabet $\Sigma$ has the *iterative* $h$-tuple property, for some $h \geq 1$, if and only if there exists an integer $p > 0$ such that for every string $z \in L$, with $|z| > p$, the following holds.

1. $z = v_1 u_1 v_2 u_2 \ldots v_h u_h v_{h+1}$, where $v_j, u_i \in \Sigma^*$ with $1 \leq i \leq h$ and $1 \leq j \leq h+1$, for some $h \geq 1$.

2. $u_1 u_2 \ldots u_h \neq \varepsilon$, i.e. not all the $u$'s are empty.

3. Posing $x = v_1 u_1^2 v_2 u_2^2 \ldots u_h^2 v_{h+1}$, there exists $z' \in L$ which is obtained from $x$ possibly permuting some substrings $u_i$ and $v_j$, with $1 \leq i < j \leq h+1$.

□

This means that any sufficiently long string $z \in L$ can be factorized into $2h+1$ substrings so that up to $h$ of them can be doubled - and suitably reordered with respect to the others - to obtain a longer string $z' \in L$. From the "pumping lemma" the following is immediate.

**Lemma 38.** *For each $n \geq 1$, any language $L \in \mathcal{L}_{D^n}$ has the iterative $2^n$-tuple property.* □

For instance, if $n = 2$ it is easy to verify that any language $L \in \mathcal{L}_{D^2}$ has the iterative 4-tuple property, by posing for each string $z$, with $|z| > p$, either $u_1 = x_{111}, u_2 = x_{112}, u_3 = w_{11}, u_4 = x_{113}, u_5 = \varepsilon, v_1 = y_{111}, v_2 = v_{11}, v_3 = u_{112}$ and $v_4 = \varepsilon$, or $u_1 = x_{111}, u_2 = w_{11}, u_3 = x_{112}x_{212}, u_4 = w_{21}, u_5 = x_{211}, v_1 = v_{11}, v_2 = u_{111}, v_3 = u_{211}$ and $v_4 = v_{21}$.

As a consequence we prove that the superfamily $\mathcal{L}_{D^+}$ forms an infinite hierarchy with respect to the parameter $n$.

**Statement 39.** For each $n \geq 1$ it holds $\mathcal{L}_{D^n} \subset \mathcal{L}_{D^{n+1}}$. □

**Proof** Obviously $\mathcal{L}_{D^n} \subseteq \mathcal{L}_{D^{n+1}}$. Now we prove that $\mathcal{L}_{D^n} \neq \mathcal{L}_{D^{n+1}}$. The language $L = \{a_1^m a_2^m \ldots a_s^m | \quad a_i \in V_T, m \geq 0 \text{ and } s = 2^n + 1\}$ is in $\mathcal{L}_{D^{n+1}}$, analogously to Example 2.2 (see also Statement 43).

We show that, from Lemma 38, it holds $L \notin \mathcal{L}_{D^n}$; in fact, choose $v$ long enough and apply the iterative $2^n$-tuple property. If for some $i$ the string $u_i$ contains two different terminals, say $u_i = \ldots a_j^h a_{j+1}^k \ldots$, with $0 < h, k \leq m$, then the string $x$ would contain the string $u_i$ twice, hence an instance of the terminal $a_j$ would occur after $a_{j+1}$. This is a contradiction.

Therefore, it must hold $u_i = a_j^k$. But $i = 1, \ldots, 2^n$, whereas $j = 1, \ldots, 2^n + 1$, so that not every index $j$ is taken into account, and we obtain a word in $L$ where the number of $a_r$'s is different from the number of $a_s$'s for some $r \neq s$, which again is a contradiction.

□

## 4 Properties of the superfamily

We briefly consider some closure properties of the superfamily

$$\mathcal{L}_{D^+} = \{L | \quad \forall n \geq 1 \; L \in \mathcal{L}_{D^n}\}$$

**Statement 40.** $\mathcal{L}_{D^+}$ is closed with respect to union. □

**Proof** Let $L_1 \in \mathcal{L}_{D^k}$ and $L_2 \in \mathcal{L}_{D^h}$. Let $t = \max(k, h)$, then $L_1 \in \mathcal{L}_{D^t}$ and $L_2 \in \mathcal{L}_{D^t}$, hence $L_1 \cup L_2 \in \mathcal{L}_{D^t}$, from Statement 26.

□

Similarly, from Statement 26 we have

**Statement 41.** $\mathcal{L}_{D^+}$ is closed with respect to catenation. □

**Statement 42.** $\mathcal{L}_{D^+}$ is closed with respect to Kleene star. □

Next we consider two operations which preserve semilinearity, namely homomorphic replication [13] and language substitution. It is well-known that the former does not preserve context-freedom.

**Statement 43.** $\mathcal{L}_{D^+}$ is closed with respect to homomorphic replication. □

**Proof** Let $L \in \mathcal{L}_{D^n}$ and let $L'$ be obtained from $L$ by means of a homomorphic replication

$$L' = \{w| \quad w = xh_1(x)\dots h_r(x) \text{ and } x \in L\}$$

where the $h_i$'s are alphabetic (and possibly erasing) homomorphisms. Then $L' \in \mathcal{L}_{D^s}$, where $s = 2r + n$.

In fact, let $M$ be a $n$-pd recognizer of $L$. The construction of an automaton $M'$ of type $PD^s$, equipped with $s$ pd tapes, accepting $L'$ is straightforward. The automaton $M'$ simulates the automaton $M$ and uses the $2r$ additional pd tapes to create $r$ homomorphic copies of the input word $x \in L$. More precisely, the $(|n|+2i-1)$-th pd tape is used to store a homomorphic reversed copy of $x$, which is then rewritten and reversed on the $(|n|+2i)$-th pd tape (without reading any input character), and so on, with $1 \leq i \leq r$. Disjoint tape symbols are used for each pd tape. The last pd tape is emptied by recognizing an input word as a homomorphic copy of the contents of the tape.

□

Since the superfamily is closed with respect to arbitrary hom., the homomorphic replication can be arbitrary, too. In [13] hom. replications are combined with string mirroring, e.g. as in $xh_1(x)h_2(x)^Rh_3(x)$; the family $\mathcal{L}_{D^n}$ is closed also with respect to such a generalization of hom. replications, as it is evident from the above proof.

A language substitution $\sigma$ transforms a language $L \subseteq \Sigma^*$ by replacing each character $b$ of $x \in L$ with a string $y = \sigma(b) \in L' \subseteq \Sigma^*$. We consider the case when both $L$ and $L'$ are mpd languages.

**Statement 44.** Take the alphabet $\Sigma = \{a_1, \dots, a_r\}$ and let $\sigma$ be a language substitution defined by the languages $L_{a_i} \in \mathcal{L}_{D^m}$. Let moreover $L \in \mathcal{L}_{D^n}$ and $L \subseteq \Sigma^*$. Then $\sigma(L) \in \mathcal{L}_{D^{m+n}}$. □

**Proof** We can simulate the recognition of $x \in L$ by means of the last $n$ tape segments. Whenever a character of $x$, say $a_i$, is to be read, the machine recognizes instead a word $y_i = \sigma(a_i) \in L_{a_i}$, using the first $m$ tapes. Since we can suppose that acceptance is by empty store (see Theorem 17), after reading $y_i$ the first $m$ tapes are empty, and the simulation of the reading of $x$ past $a_i$ can continue.

□

*Note 45.* From Statement 44 it follows that every family $\mathcal{L}_{D^n}$ is closed with respect to arbitrary, erasing homomorphism.

**Theorem 46.** $\mathcal{L}_{D^+}$ *is an Abstract Family of Languages* (AFL) [2]. □

The statement follows immediately from the previous results. Finally, we prove that not all semilinear languages are multi-depth.

**Statement 47.** The superfamily $\mathcal{L}_{D^+}$ is strictly contained in the family of context-sensitive semilinear languages (in the sense of Parikh [19]). □

**Proof** From Statement 20 multi-depth languages are semilinear. Consider the semilinear language $L = \{a^{f(m)}b^m c^{m-f(m)} | \quad f : N \rightarrow N$ is a non-linear, but computable, integer function, such that $f(m) \leq m$ for any $m \geq 0\}$[3]. There exists no integer $n \geq 1$ such that a grammar $G \in \mathcal{G}_{D^n}$ generates $L$. In fact, Statements 24, 25 and 27, Note 45 and Nivat theorem [16] imply that $\mathcal{G}_{D^n}$ is closed with respect to rational transduction. Imagine now a rational transduction $\tau$ that deletes the prefix of the strings of $L$ containing $a$'s and $b$'s; the rest of the string is left unalterated. Then the transduced language $\tau(L) = \{c^{m-f(m)} | \quad m \geq 0\}$ is (obviously) not semilinear, hence $L$ is not in $\mathcal{L}_{D^n}$, for any $n \geq 1$.

□

## 5 Conclusions

For determinism [20], an aspect not covered here, the strict inclusion of deterministic by non-deterministic languages (as for cf languages) is proved in the related paper [21]. More precisely, it is proved that the deterministic 2-pd languages are strictly included by the 2-pd (non-deterministic) ones, and that there are (non-deterministic) cf languages which are not 2-pd deterministic.

Another important property of cf languages which is preserved by mpd languages concerns the complexity of recognition, which remains polynomial-time [10]. Practical linear-time top-down parsing algorithms for the deterministic case have been investigated [7, 18], in view of the definition of a subfamily enjoying the same advantages as the $LL(k)$ cf languages [14]. This development would allow a straightforward extension of the classical parser generating tools, opening the door to the exploitation of $D^n$ grammars for compiler writing.

On the conceptual side we conclude by observing that mpd automata use for their store an array of LIFO tapes. More general data structures have been considered for the store, such as an array of FIFO or LIFO tapes. A corresponding class of grammars was defined in the same spirit of mpd grammars and investigated in [4, 6, 7]. It includes some types of queue [1, 9] and dequeue automata [8] (see also [3]). Not all the properties of mpd languages remain valid for this more general class: e.g. that of being an AFL [4] is lost.

Acknowledgment Thanks to Dino Mandrioli and to Pierluigi Sanpietro for their help. An anonymous referee has given valuable suggestions for improving the presentation.

## References

1. E. Allevi, A. Cherubini, S. Crespi Reghizzi, "Breadth-first Phrase-Structure Grammars and Queue Automata", *L. Notes in Comput. Sci.* **324** (1988) pp. 162-170.

[3] For instance, the assignment $m \stackrel{f}{\mapsto} |\sqrt{m}|$ might do.

2. J. Berstel, Transductions and Context-free Languages, (Teubner Studienbücher, 1979).
3. F. J. Brandenburg, "On the Intersections of Stacks and Queues", *Theoret. Comput. Sci.* **58**, (1988) pp. 69-80.
4. L. Breveglieri, A. Cherubini, C. Citrini, S. Crespi Reghizzi, "Stacks, Queues and their Languages", Int. Rep. n. 90-053, Dip. Elettronica e Informazione, Politecnico di Milano, (1990).
5. L. Breveglieri, A. Cherubini, C. Citrini, S. Crespi Reghizzi, "Multistack Languages and Grammars", Int. Rep. n. 93-018, Dip. di Elettronica e Informazione, Politecnico di Milano, (1993).
6. L. Breveglieri, A. Cherubini, S. Crespi Reghizzi, "A Chomsky-Schützenberger Property for generalized Context-free Grammars", Int. Rep. n. 93-062, Dip. di Elettronica e Informazione, Politecnico di Milano, (1993).
7. L. Breveglieri, A. Cherubini, S. Crespi Reghizzi, "Deterministic Parsing for augmented Context-free Grammars", *L. Notes in Comput. Sci.* **969** (1995) pp. 326-336.
8. A. Cherubini, C. Citrini, S. Crespi Reghizzi, D. Mandrioli, "Breadth and Depth Grammars and Deque Automata", *Int. Jou. Found. Comput. Sci.* n. **1** 1990 pp. 219-232.
9. A. Cherubini, C. Citrini, S. Crespi Reghizzi, D. Mandrioli, "QRT FIFO Automata, Breadth-first Grammars and their Relations", *Theoret. Comput. Sci.* **85** 1991 pp. 171-203.
10. A. Cherubini, P. Sanpietro, "A polynomial-Time parsing Algorithm for $k$-Depth Languages", *Jou. of Comput. Sys. Sci.*, to appear
11. N. Chomsky, "Context-free Grammars and push-down Storages", Quart. Prog. Rep. n. 65, MIT Res. Lab. Elect., MA, (1962).
12. J. Dassow, G. Păun, "*Regulated Rewriting in formal Language Theory*", (EATCS Monograph Series n. 18, Springer Verlag, 1989).
13. S. Ginsburg, E. H. Spanier, "AFL with the semilinear Property", *Jou. Comput. Sys. Sci.* **5** 1971 pp. 365-396
14. M. A. Harrison, Introduction to formal Language Theory, (Addison Wesley 1978).
15. Z. Manna, The mathematical Theory of Computation, (McGraw Hill 1974).
16. M. Nivat, "Transductions des Languages de Chomsky", Thèse d'Etat, Université de Paris, 1967, or Annales Institut Fourier n. 18, pp. 339-456, (1968).
17. W. Ogden, "A helpful Result for proving inherent Ambiguity", in *Mathematical Systems Theory*, vol. 2, n° 3, pp. 191-194, (1968).
18. R. Pelizzoli, "Analisi sintattica deterministica con Grammatiche lineari estese" (Syntactic deterministic Analysis of linear extended Grammars), Thesis, Dip. di Scienze dell'Informazione, Università di Milano, 1994.
19. R. J. Parikh, "Language generating Devices", Quart. Prog. Rep. n. 60, MIT Res. Lab. Elect., pp. 191-194, (1961).
20. A. Salomaa, Formal Languages, (ACM Monograph Series, Academic Press, 1973)
21. P. San Pietro, "Two Stack Automata", Int. Rep. n. 92-073, Dip. di Elettronica e Informazione, Politecnico di Milano, Milano, Italy, 1992

## A Proof of Lemma 16

First we shall prove that if a language $L$ is recognized by empty tape by a $n$-pd automaton $M$, then $L = L(G)$, where $G \in \mathcal{G}_{D^n}$. Let $L = L(M)$, where $M$ is a $n$-pd automaton in (strong) normal form (Note 15). We shall construct a grammar $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$ in this way:

$$V_N = \{\Gamma \times Q^{2n}\} \uplus \{\Gamma \times Q^2\} \qquad V_T = \Sigma \qquad S = Z_0$$

and the productions in $P$ have the form:

1. $\forall q \in Q$, and $Z_0 \rightarrow \varepsilon$ iff $\varepsilon \in L$

$$Z_0 \rightarrow (\langle Z_0; q_0, q; q, q; \ldots; q, q\rangle)_1$$

2. $\forall q'_{2h} \in Q$ iff $(q'_1, B^{(1)}C^{(1)}, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(1)})$, for $2 \leq h \leq n$

$$\langle A^{(1)}; q_1, q_2; q_3, q_4; \ldots; q_{2n-1}, q_{2n}\rangle \rightarrow$$
$$(\langle B^{(1)}; q'_1, q'_2; q_3, q'_4; \ldots; q_{2n-1}, q'_{2n}\rangle \langle C^{(1)}; q'_2, q_2; q'_4, q_4; \ldots; q'_{2n}, q_{2n}\rangle)_1$$

3. iff $(q_2, \varepsilon, \ldots, B^{(j)}, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(1)})$

$$\langle A^{(1)}; q_1, q_2; q_3, q_3; \ldots; q_{2j-3}, q_{2j-3}; q_{2j-1}, q_{2j}; q_{2j+1}, q_{2j+1}; \ldots; q_{2n-1}, q_{2n-1}\rangle$$
$$\rightarrow (\langle B^{(j)}; q_{2j}, q_{2j-1}\rangle)_j$$

4. $\forall q, q_{2h+1} \in Q$, for $1 \leq h \leq j-1$) iff $(q'_1, C^{(1)}, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(j)})$

$$\langle A^{(j)}; q_1, q_2\rangle \rightarrow (\langle C^{(1)}; q'_1, q; q_3, q; \ldots; q_{2j-3}, q; q_2, q; q_1, q; \ldots; q_1, q\rangle)_1$$

5. iff $(q_2, \varepsilon, \ldots, \varepsilon) \in \delta(q_1^{(1)}, a, A^{(1)})$

$$\langle A^{(1)}; q_1, q_2; q_3, q_3; \ldots; q_{2n-1}, q_{2n-1}\rangle \rightarrow a$$

We shall prove that $w \in L(G)$ if and only if $w \in L(M)$. The grammar $G$ simulates the behaviour of $M$ by guessing (in a non-deterministic way) the correct sequence of states, coded in the first and $(2h+2)$-th state component of the first pd tape symbols ($h > 1$). We shall prove that when a prefix $v$ of $w$ has been rewritten by $G$, the non-terminal string (sentential form) has one of the following structures, where the most meaningful equalities have been highlighted:

$$\begin{aligned}
&(\langle A^{(1)}; q_1, \mathbf{q_2}; \underline{p_1}, \mathbf{p_2}; \ldots; \underline{r_1}, \mathbf{r_2}\rangle \langle B^{(1)}; \mathbf{q_2}, \mathbf{q_3}; \mathbf{p_2}, \mathbf{p_3}; \ldots; \mathbf{r2}, \mathbf{r3}\rangle \\
&\langle C^{(1)}; \mathbf{q_3}, \mathbf{q_4}; \mathbf{p_3}, \mathbf{p_4}; \ldots; \mathbf{r_3}, \mathbf{r_4}\rangle \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (7)\\
&\ldots\ldots\ldots\ldots\ldots\ldots \langle H^{(1)}; \mathbf{q_h}, q_{h+1}; \mathbf{p_h}, q_{h+1}; \ldots; \mathbf{r_h}, q_{h+1}\rangle)_1 \\
&(\langle A^{(2)}; \underline{p_1}, p'_2\rangle \ldots \langle K^{(2)}; p'_k, p'_{k+1}\rangle)_2 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&(\langle A^{(n)}; \underline{r_1}, r'_2\rangle \ldots \langle M^{(n)}; r'_m, r'_{m+1}\rangle)_n
\end{aligned}$$

or

$$
\begin{array}{l}
(\langle A^{(j)}; \mathbf{q}, p_2\rangle \ldots \langle K^{(j)}; p_k, p_{k+1}\rangle)_j \\
(\langle A^{(j+1)}; \mathbf{q}, s_2\rangle \ldots \langle L^{(j+1)}; s_l, s_{l+1}\rangle)_{j+1} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle A^{(n)}; \mathbf{q}, r_2\rangle \ldots \langle M^{(n)}; r_m, r_{m+1}\rangle)_n
\end{array} \tag{8}
$$

with $j \geq 2$. In both forms some segment may be empty. The corresponding configuration of the automaton is

$$\langle q_1, x; A^{(1)}B^{(1)}\ldots H^{(1)}, A^{(2)}B^{(2)}\ldots K^{(2)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

or

$$\langle q, x; \varepsilon, \ldots, \varepsilon, A^{(j)}B^{(j)}\ldots K^{(j)}, A^{(j+1)}B^{(j+1)}\ldots L^{(j+1)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

with $w = vx$. We show this fact by induction. The form of production 1. implies that the first state is the initial state $q_0$ and the topmost state of the $i$-th pd tape ($q'$) is recorded into the $(2i-1)$-th state component of the top symbol. Moreover, the empty word is rewritten by $G$ and the configuration is $\langle q_0, x; Z_0, \varepsilon, \varepsilon, \ldots, \varepsilon\rangle$. So the base of the induction is stated.

Let us now show that productions from 2. to 5. change the forms (7) and (8) of the store into themselves and satisfy our conditions. Applying production 2., written as

$$\forall q'', p'', \ldots, r'' \in Q, \text{ iff } (q', X^{(1)}Y^{(1)}, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(1)})$$

$$
\begin{array}{l}
\langle A^{(1)}; q_1, q_2; p_1, p_2; \ldots; r_1, r_2\rangle \rightarrow \\
(\langle X^{(1)}; q', q''; p_1, p''; \ldots; r_1, r''\rangle\langle Y^{(1)}; q'', q_2; p'', p_2; \ldots; r'', r_2\rangle)_1
\end{array}
$$

to the configuration (7), we obtain

$$
\begin{array}{l}
(\langle X^{(1)}; q', q''; p_1, p''; \ldots; r_1, r''\rangle\langle Y^{(1)}; q'', q_2; p'', p_2; \ldots; r'', r_2\rangle \\
\ \langle B^{(1)}; q_2, q_3; p_2, p_3; \ldots; r_2, r_3\rangle\langle C^{(1)}; q_3, q_4; p_3, p_4; \ldots; r_3, r_4)\rangle \\
\ \ldots\ldots\ldots\ldots\ldots\ldots\langle H^{(1)}; q_h, q_{h+1}; p_h, q_{h+1}; \ldots; r_h, q_{h+1}\rangle)_1 \\
(\langle A^{(2)}; p_1, p_2'\rangle \ldots \langle K^{(2)}; p_k', p_{k+1}'\rangle)_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle A^{(n)}; r_1^{(n)}, r_2'\rangle \ldots \langle M^{(n)}; r_m', r_{m+1}'\rangle)_n
\end{array}
$$

which has the same form as (7). The new state is $q'$. Moreover, by the induction hypothesis the configuration of $M$ was

$$\langle q_1, x; A^{(1)}B^{(1)}\ldots H^{(1)}, A^{(2)}B^{(2)}\ldots K^{(2)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

with $w = vx$, and the transition is

$$
\begin{array}{l}
\langle q_1, x; A^{(1)}B^{(1)}\ldots H^{(1)}, A^{(2)}B^{(2)}\ldots K^{(2)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle \vdash_A \\
\langle q', x; X^{(1)}Y^{(1)}B^{(1)}\ldots H^{(1)}, A^{(2)}B^{(2)}\ldots K^{(2)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle
\end{array}
$$

Applying production 3., written as

$$\langle A^{(1)}; q_1, q_2; p_1, p_1; \ldots; m_1, m_1; s_1, s_2; t_1, t_1; \ldots; r_1, r_1\rangle \rightarrow (\langle X^{(j)}; s_2, s_1\rangle)_j$$

iff $(q_2, \varepsilon, \ldots, X^{(j)}, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(1)})$, to the configuration (7), we obtain

$$\begin{array}{l}
(\langle B^{(1)}; q_2, q_3; p_1, p_3; \ldots; m_1, m_3; s_2, s_3; t_1, t_3; \ldots; r_1, r_3\rangle \\
\ \langle C^{(1)}; q_3, q_4; p_3, p_4; \ldots; r_3, r_4\rangle \ldots\ldots\ldots\ldots\ldots\ldots \\
\ \ldots\ldots\ldots\ldots\ldots\ldots \langle H^{(1)}; q_h, q_{h+1}; p_h, q_{h+1}; \ldots; r_h, q_{h+1}\rangle)_1 \\
(\langle A^{(2)}; p_1, p_2'\rangle \ldots \langle K^{(2)}; p_k', p_{k+1}'\rangle)_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle X^{(j)}; s_2, s_1\rangle \langle A^{(j)}; s_1, s_2'\rangle \ldots \langle L^{(j)}; s_l', s_{l+1}'\rangle)_j \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle A^{(n)}; r_1^{(n)}, r_2'\rangle \ldots \langle M^{(n)}; r_m', r_{m+1}'\rangle)\rangle)_n
\end{array}$$

which has the same form, for $h > 1$. The state becomes $q_2$. The new configuration is now

$$\langle q_2, x; B^{(1)} \ldots H^{(1)}, A^{(2)} B^{(2)} \ldots K^{(2)}, \ldots, X^{(j)} A^{(j)} \ldots L^{(j)}, \ldots, A^{(n)} B^{(n)} \ldots M^{(n)}\rangle$$

For $h = 1$ the form (7) was

$$\begin{array}{l}
(\langle A^{(1)}; q_1, q_2; p_1, q_2; \ldots; r_1, q_2\rangle)_1 \\
(\langle A^{(2)}; p_1, p_2'\rangle \ldots \langle K^{(2)}; p_k', p_{k+1}'\rangle)_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle A^{(n)}; r_1, r_2'\rangle \ldots \langle M^{(n)}; r_m, r_{m+1}'\rangle)_n
\end{array}$$

where, in order to apply production 3., we have to suppose that for $i \neq 2, 3, 2j, 2j+1$, all the states coded in the $i$-th component of the first pd store symbols are equal to $q_2$. Hence this form becomes

$$\begin{array}{l}
(\langle A^{(2)}; q_2, p_2'\rangle \ldots \langle K^{(2)}; p_k', p_{k+1}'\rangle)_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle X^{(j)}; q_2, s_1\rangle \langle A^{(j)}; s_1, s_2'\rangle \ldots \langle L^{(j)}; s_l', s_{l+1}'\rangle)_j \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\langle A^{(n)}; q_2, r_2'\rangle \ldots \langle M^{(n)}; r_m', r_{m+1}'\rangle)_n
\end{array}$$

which has form (8). The new configuration of the automaton is now

$$\langle q_2, x; A^{(2)} B^{(2)} \ldots K^{(2)}, \ldots, X^{(j)} A^{(j)} B^{(j)} \ldots L^{(j)}, \ldots, A^{(n)} B^{(n)} \ldots M^{(n)}\rangle$$

Applying production 4., written as

$$\forall q, q_{2h-1} \in Q \text{ for } (2 \leq h \leq j-1), \quad \text{iff } (q_1', X^{(1)}, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, \varepsilon, A^{(j)})$$

$$\langle A^{(j)}; q_1, q_2\rangle) \rightarrow (\langle X^{(1)}; q_1', q; q_3, q; \ldots; q_{2j-3}, q; q_2, q; q_1, q; \ldots; q_1, q\rangle)_1$$

to the configuration (8), which obviously has the form

$$\begin{array}{l}(\langle A^{(j)}; q_1, q_2\rangle\langle B^{(j)}; q_2, s_3'\rangle \ldots \langle L^{(j)}; s_l', s_{l+1}'\rangle)_j\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ (\langle A^{(n)}; q_1, r_2'\rangle \ldots \langle M^{(n)}; r_m', r_{m+1}'\rangle)_n\end{array}$$

we obtain

$$\begin{array}{l}(\langle X^{(1)}; q_1', q; q_3, q; \ldots; q_{2j-3}, q; q_2, q; q_1, q; \ldots; q_1, q\rangle)_1\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ (\langle B^{(j)}; q_2, s_3^{(j)}\rangle \ldots \langle L^{(j)}; s_l, s_{k+1}^{(j)}\rangle)_j\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ (\langle A^{(n)}; q_1, r_2'\rangle\langle B^{(n)}; r_2', r_3'\rangle \ldots \langle M^{(n)}; r_m', r_{m+1}'\rangle)_n\end{array}$$

which has form (7). The state becomes $q_1'$. The configuration of $M$ was

$$\langle q_1, x; \varepsilon, \ldots, A^{(j)}B^{(j)}\ldots L^{(j)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

the new configuration becomes

$$\langle q_1', x; X^{(1)}, \varepsilon, \ldots, A^{(j)}B^{(j)}\ldots L^{(j)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

Applying production 5., written as

$$\langle A^{(1)}; q_1, q_2; q_3, q_3; \ldots; q_{2n-1}, q_{2n-1}\rangle \to a \quad \text{iff } (q_2, \varepsilon, \ldots, \varepsilon) \in \delta(q_1, a, A^{(1)})$$

to the configuration (7), with $h > 1$, we obtain

$$\begin{array}{l}(\langle B^{(1)}; q_2, q_3'; q_3, p_3'; \ldots; q_{2n-1}, r_3'\rangle\langle C^{(1)}; q_3', q_4'; p_3', p_4'; \ldots; r_3', r_4'\rangle\\ \quad\ldots\ldots\ldots\ldots\ldots\ldots\langle H^{(1)}; q_h', q_{h+1}; p_h', q_{h+1}; \ldots; r_h', q_{h+1}\rangle)_1\\ (\langle A^{(2)}; q_3, p_2\rangle\langle B^{(2)}; p_2, p_3\rangle \ldots \langle K^{(2)}; p_k, p_{k+1}\rangle)_2\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ (\langle A^{(n)}; q_1, r_2\rangle\langle B^{(n)}; r_2, r_3\rangle \ldots \langle M^{(n)}; r_m, r_{m+1}\rangle)_n\end{array}$$

which has the same form as (7). The state becomes $q_2$. The rewritten terminal string is $va$. The configuration of $M$ was

$$\langle q_1, ay; A^{(1)}B^{(1)}\ldots H^{(1)}, A^{(2)}B^{(2)}\ldots K^{(2)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

the new configuration becomes

$$\langle q_2, y; B^{(1)}\ldots H^{(1)}, A^{(2)}B^{(2)}\ldots K^{(2)}, \ldots, A^{(n)}B^{(n)}\ldots M^{(n)}\rangle$$

If $h = 1$, the form (7) was

$$\begin{array}{l}(\langle A^{(1)}; q_1, q_2; q_3, q_2; \ldots; q_{2n-1}, q_2\rangle)_1\\ (\langle A^{(2)}; q_3, p_2\rangle \ldots \langle K^{(2)}; p_k, p_{k+1}\rangle)_2\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ (\langle A^{(n)}; q_{2n-1}, r_2\rangle \ldots \langle M^{(n)}; r_m, r_{m+1}\rangle)_n\end{array}$$

where we have $q_2 = q_3 = \ldots = q_{2n-1}$, hence it becomes the sentence

$$(\langle A^{(2)}; q_2, p_2\rangle \ldots \langle K^{(2)}; p_k, p_{k+1}\rangle)_2 \ldots (\langle A^{(n)}; q_2, r_2\rangle \ldots \langle M^{(n)}; r_m^{(n)}, r_{m+1}^{(n)}\rangle)_n$$

The rewritten terminal string is $va$. The configuration of $M$ was

$$\langle q_1, ay; A^{(1)}, \ldots, A^{(n)}B^{(n)} \ldots M^{(n)}\rangle$$

the new configuration becomes

$$\langle q_2, y; \varepsilon, A^{(2)} \ldots K^{(2)}, \ldots, A^{(n)}B^{(n)} \ldots M^{(n)}\rangle$$

Hence, when the non-terminal string has been completely rewritten, the tape of $M$ is empty and $w$ is recognized by $M$. Analogously, we can prove that if a configuration of $M$ is

$$\langle q_1, x; A^{(1)}B^{(1)} \ldots H^{(1)}, A^{(2)}B^{(2)} \ldots K^{(2)}, \ldots, A^{(n)}B^{(n)} \ldots M^{(n)}\rangle$$

when a prefix $v$, with $w = vx$, is scanned by $M$, the derivation

$$\begin{aligned} Z_0 \stackrel{*}{\Rightarrow} w(&\langle A^{(1)}; q_1^{(1)}, q_2^{(1)}; q_1^{(2)}, q_2^{(2)}; \ldots; q_1^{(n)}, q_2^{(n)}\rangle \\ &\langle B^{(1)}; q_2^{(1)}, q_3^{(1)}; q_2^{(2)}, q_3^{(2)}; \ldots; q_2^{(n)}, q_3^{(n)}\rangle \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &\langle H^{(1)}; q_h^{(1)}, q_{h+1}; q_h^{(2)}, q_{h+1}; \ldots; q_h^{(n)}, q_{h+1}\rangle)_1 \\ &(\langle A^{(2)}; q_1^{(2)}, p_2^{(2)}\rangle\langle B^{(2)}; p_2^{(2)}, p_3^{(2)}\rangle \ldots \langle K^{(2)}; p_k^{(2)}, p_{k+1}^{(2)}\rangle)_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &(\langle A^{(n)}; q_1^{(n)}, r_2^{(n)}\rangle\langle B^{(n)}; r_2^{(n)}, r_3^{(n)}\rangle \ldots \langle M^{(n)}; r_m^{(n)}, q'^{(n)}\rangle)_n \end{aligned}$$

is a derivation of $G$. So our claim is proved.

Conversely, given $G = (V_N, V_T, P, S) \in \mathcal{G}_{D^n}$, from Statement 10 we can assume that $G$ has productions of the type $A \rightarrow (\alpha_1)_1(\alpha_2)_2 \ldots (\alpha_n)_n$, with $\alpha_i \in V_N^*$, and of the type $A \rightarrow a$, with $a \in V_T$. We construct the $n$-pd automaton $M = (q_0, \Sigma, \Gamma, \delta, q_0, Z_0)$, with only one state, as specified in the following:

$$\Sigma = V_T \qquad \Gamma = V_N \qquad Z_0 = S$$

$$(\alpha_1, \ldots, \alpha_n) \in \delta(\varepsilon, A) \qquad \text{iff } A \rightarrow (\alpha_1)_1 \ldots (\alpha_n)_n \in P$$

$$(\varepsilon, \ldots, \varepsilon) \in \delta(a, A) \qquad \text{iff } A \rightarrow a \in P$$

where $A \in V_N$, $a \in V_T$ and $\alpha_j \in V_N^*$ for any $1 \leq j \leq n$. The proof that $L(M) = L(G)$ is straightforward and therefore is omitted.

## B Proof of the Pumping Lemma

For simplicity we consider the case $n = 3$. Let $L \in \mathcal{L}_{D^3}$, i.e. let $L = L(G)$ with $G \in \mathcal{G}_{D^3}$, and suppose that $G = (V_N, V_T, P, S)$ is in strong normal form (see Note 14). This means that $V_N$ is the disjoint union of $V_N^{(1)} \uplus V_N^{(2)} \uplus V_N^{(3)}$ and that the productions have the following forms:

$$P = \begin{cases} A^{(i)} \to (A^{(1)})_1 & \text{for } i = 2,3 \\ A^{(1)} \to (B^{(1)}C^{(1)})_1 \\ A^{(1)} \to (B^{(2)})_2 \\ A^{(1)} \to (B^{(3)})_3 \\ A^{(1)} \to a \end{cases}$$

If a word $z \in L$ is long enough, then in its syntax tree there exists a path where a non-terminal $A$ occurs twice; we can suppose - from the form of $P$ - that this non-terminal belongs to $V_N^{(1)}$. Denote by $A_1^{(1)}$ its occurrence closest to the root of the tree and by $A_2^{(1)}$ the following one.

Among such paths we can choose $A_1^{(1)}$ and $A_2^{(1)}$ so that in the path $[A_1^{(1)}, A_2^{(1)}]$ there exists at least one bifurcation. Otherwise we could delete all the paths of the kind $[A_1^{(1)}, A_2^{(1)}]$ obtaining a syntax tree of a permutation $z''$ of $z$, with $z'' \in L$, containing no path of length exceeding $n + 2$.

Furthermore, we can choose the path $[A_1^{(1)}, A_2^{(1)}]$ in such a way that it is the most distant from the root, among the paths satisfying the previous conditions. Now we distinguish some cases.

Case 1 In the path $[A_1^{(1)}, A_2^{(1)}]$ there exists no node belonging to $V_N^{(2)} \cup V_N^{(3)}$, which means that:

$$\begin{aligned} S &\overset{+}{\Rightarrow}_G x_{11}(A_1^{(1)}\beta_{11})_1(\xi_{21})_2(\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}(A_2^{(1)}\gamma_{11}\beta_{11})_1(\nu_{21}\xi_{21})_2(\nu_{31}\xi_{31})_3 \end{aligned}$$

where $x_{11}, v_{11} \in V_T^*$, $A_1^{(1)}, A_2^{(1)} \in V_N^{(1)}$, $\gamma_{11}, \beta_{11} \in (V_N^{(1)})^*$, $\nu_{21}, \xi_{21} \in (V_N^{(2)})^*$ and $\nu_{31}, \xi_{31} \in (V_N^{(3)})^*$. Hence:

$$\begin{aligned} S &\overset{+}{\Rightarrow}_G x_{11}v_{11}(A_2^{(1)}\gamma_{11}\beta_{11})_1(\nu_{21}\xi_{21})_2(\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}(\gamma_{11}\beta_{11})_1(\alpha_{21}\nu_{21}\xi_{21})_2(\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}(\beta_{11})_1(\gamma_{21}\alpha_{21}\nu_{21}\xi_{21})_2(\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}(\beta_{21}\gamma_{21}\alpha_{21}\nu_{21}\xi_{21})_2(\beta_{31}\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}t_{21}(\gamma_{21}\alpha_{21}\nu_{21}\xi_{21})_2(\beta_{32}\beta_{31}\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}t_{21}u_{21}(\alpha_{21}\nu_{21}\xi_{21})_2(\gamma_{32}\beta_{32}\beta_{31}\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}t_{21}u_{21}w_{21}(\nu_{21}\xi_{21})_2(\alpha_{32}\gamma_{32}\beta_{32}\beta_{31}\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}t_{21}u_{21}w_{21}v_{21}(\xi_{21})_2(\nu_{32}\alpha_{32}\gamma_{32}\beta_{32}\beta_{31}\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}t_{21}u_{21}w_{21}v_{21}x_{21}(\xi_{32}\nu_{32}\alpha_{32}\gamma_{32}\beta_{32}\beta_{31}\gamma_{31}\alpha_{31}\nu_{31}\xi_{31})_3 \\ &\overset{+}{\Rightarrow}_G x_{11}v_{11}w_{11}u_{11}t_{11}t_{21}u_{21}w_{21}v_{21}x_{21}x_{32}v_{32}w_{32}u_{32}t_{32}t_{31}u_{31}w_{31}v_{31}x_{31} \end{aligned}$$

Then, posing the strings

$$\begin{aligned} z_{11} &= x_{11}v_{11}w_{11}u_{11}t_{11} \\ z_{21} &= x_{21}v_{21}w_{21}u_{21}t_{21} \\ z_{31} &= x_{31}v_{31}w_{31}u_{31}t_{31} \\ z_{32} &= x_{32}v_{32}w_{32}u_{32}t_{32} \end{aligned}$$

and $z_1 = z_{11}$, we have $z = z_1 z_2 z_3$, with $z_2 = z_{21}^R$ and $z_3 = z_{32} z_{31}^R$. In this derivation we have used the following partial derivations:

$$\begin{aligned} A_1^{(1)} &\stackrel{+}{\Rightarrow}_G v_{11}(A_2^{(1)}\gamma_{11})_1(\nu_{21})_2(\nu_{31})_3 \\ &\stackrel{+}{\Rightarrow}_G v_{11}w_{11}(\gamma_{11})_1(\alpha_{21}\nu_{21})_2(\alpha_{31}\nu_{31})_3 \\ &\stackrel{+}{\Rightarrow}_G v_{11}w_{11}u_{11}(\gamma_{21}\alpha_{21}\nu_{21})_2(\gamma_{31}\alpha_{31}\nu_{31})_3 \\ &\stackrel{+}{\Rightarrow}_G \ldots \end{aligned} \tag{9}$$

and

$$\begin{aligned} A_2^{(1)} &\stackrel{+}{\Rightarrow}_G w_{11}(\alpha_{21})_2(\alpha_{31})_3 \\ &\stackrel{+}{\Rightarrow}_G w_{11}w_{21}(\alpha_{32}\alpha_{31})_3 \\ &\stackrel{+}{\Rightarrow}_G w_{11}w_{21}w_{31} \end{aligned} \tag{10}$$

Hence, if in the sentential form $x_{11}v_{11}(A_2^{(1)}\gamma_{11}\beta_{11})_1(\nu_{21}\xi_{21})_2(\nu_{31}\xi_{31})_3$ we use the derivation (9) (recall that $A_2^{(1)} = A_1^{(1)}$), we have:

$$\begin{aligned} S &\stackrel{+}{\Rightarrow}_G x_{11}v_{11}^2(A_2^{(1)}\gamma_{11}^2\beta_{11})_1(\nu_{21}^2\xi_{21})_2(\nu_{31}^2\xi_{31}) \\ &\stackrel{+}{\Rightarrow}_G x_{11}v_{11}^2w_{11}u_{11}^2t_{11}t_{21}u_1^2w_{21}v_{21}^2x_{21}x_{32}v_{32}^2w_{32}u_{32}^2t_{32}t_{31}u_{31}^2w_{31}v_{31}^2x_{31}v_{ij} \end{aligned}$$

Hence, posing the lists

$$\overline{x}_{ij} = (x_{ij})(t_{ij}), \quad \overline{y}_{ij} = (\varepsilon) \quad \text{and} \quad \overline{u}_{ij} = (u_{ij})$$

and the strings

$$\begin{aligned} z_{ij} &= m(\overline{x}_{ij}, \overline{y}_{ij}v_{ij}w_{ij}\overline{u}_{ij}) = x_{ij}v_{ij}w_{ij}u_{ij}t_{ij} \\ z'_{ij} &= m(\overline{x}_{ij}, v_{ij}\overline{y}_{ij}v_{ij}w_{ij}u_{ij}\overline{u}_{ij}) = x_{ij}v_{ij}^2w_{ij}u_{ij}^2t_{ij} \\ z'_1 &= z'_{11}, \quad z'_2 = {z'_{21}}^R \quad \text{and} \quad z'_3 = z'_{32}{z'_{31}}^R \end{aligned}$$

it follows

$$S \stackrel{+}{\Rightarrow}_G z' = z'_1 z'_2 z'_3$$

i.e. conditions (4) and (6) hold. Finally, condition (5) follows from the form of $P$ and from the consideration on the subtree of $T'$ with root $A_1^{(1)}$, posing $q = 2^s$ and $s = |V_N^{(1)}| + 1$.

Case 2 In the path from $A_1^{(1)}$ to $A_2^{(1)}$ there exists at least one node labeled by a symbol which is in $V_N^{(2)}$, and no nodes labeled by a symbol in $V_N^{(3)}$. This means that:

$$
\begin{aligned}
S &\stackrel{+}{\Rightarrow}_{G'} x_{11}(A_1^{(1)}\beta_{11})_1(\xi_{21})_2(\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}(\beta_{11})_1(\eta_{21}\xi_{21})_2(\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}(\beta_{21}\eta_{21}\xi_{21})_2(\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}(\eta_{21}\xi_{21})_2(\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}(B^{(2)}\eta'_{21}\xi_{21})_2(\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}(A_2^{(1)})_1(\eta'_{21}\xi_{21})_2(\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}w_{11}(\alpha_{21}\eta'_{21}\xi_{21})_2(\alpha_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}w_{11}w_{21}(\eta'_{21}\xi_{21})_2(\alpha_{32}\alpha_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}w_{11}w_{21}y'_{21}(\xi_{21})_2(\eta'_{32}\alpha_{32}\alpha_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}w_{11}w_{21}y'_{21}x_{21}(\xi_{32}\eta'_{32}\alpha_{32}\alpha_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}w_{11}w_{21}y'_{21}x_{21}x_{32}(\eta'_{32}\alpha_{32}\alpha_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_{G'} x_{11}y_{11}t_{11}t_{21}v_{21}w_{11}w_{21}y'_{21}x_{21}x_{32}y'_{32}w_{32}w_{31}v_{32}t_{32}t_{31}y_{31}x_{31}
\end{aligned}
$$

where we have derived $A_1^{(1)}$ as

$$
\begin{aligned}
A_1^{(1)} &\stackrel{+}{\Rightarrow}_G y_{11}(\eta_{21})_2(\eta_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G y_{11}v_{21}(B^{(2)}\eta'_{21})_2(\nu_{32}\eta_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G y_{11}v_{21}(A_2^{(1)})_1(\eta'_{21})_2(\nu_{32}\eta_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G y_{11}v_{21}w_{11}(\alpha_{21}\eta'_{21})_2(\alpha_{31}\nu_{32}\eta_{31})_3 \qquad (11) \\
&\stackrel{+}{\Rightarrow}_G y_{11}v_{21}w_{11}w_{21}(\eta'_{21})_2(a_{32}a_{31}\nu_{32}\eta_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G y_{11}v_{21}w_{11}w_{21}y'_{21}(\eta'_{32}\alpha_{32}\alpha_{31}\nu_{32}\eta_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G y_{11}v_{21}w_{11}w_{21}y'_{21}y'_{32}w_{32}w_{31}v_{32}y_{31}
\end{aligned}
$$

and $A_2^{(1)}$ as

$$
\begin{aligned}
A_2^{(1)} &\stackrel{+}{\Rightarrow}_G w_{11}(\alpha_{21})_2(\alpha_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G w_{11}w_{21}(\alpha_{32}\alpha_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G w_{11}w_{21}w_{32}w_{31}
\end{aligned}
$$

Hence, if we use derivation (11) instead of the above one to derive $A_2^{(1)}$, we get:

$$
\begin{aligned}
S &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}v_{21}(A_2^{(1)})_1(\eta'_{21}\xi_{21})_2(\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3 \\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}v_{21}y_{11}(\eta_{21}\eta'_{21}\xi_{21})_2(\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3
\end{aligned}
$$

$$
\begin{aligned}
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_{11}v_{21}(B^{(2)}\eta'_{21}\xi_{21})_2(\nu_{32}\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_1v_{21}(A_2^{(1)})_1(\eta'_{21}\xi_{21})_2(\nu_{32}\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_1v_{21}w_{11}(\alpha_{21}\eta'_{21}\xi_{21})_2(\alpha_{31}\nu_{32}\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_1v_{21}w_{11}w_{21}(\eta'_{21}\xi_{21})_2(\alpha_{32}\alpha_{31}\nu_{32}\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_1v_{21}w_{11}w_{21}y'_{21}(x_{21})_2(\eta'_{32}\alpha_{32}\alpha_{31}\nu_{32}\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_1v_{21}w_{11}w_{21}y'_{21}x_{21}(x_{32}\eta'_{32}\alpha_{32}\alpha_{31}\nu_{32}\eta_{31}\nu_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}v_{21}y_1v_{21}w_{11}w_{21}y'_{21}x_{21}x_{32}y'_{32}w_{32}w_{31}v_{32}y_{31}v_{32}t_{32}t_{31}y_{31}x_{31}
\end{aligned}
$$

Hence, we have that if $z = z_1z_2 \in L$, with $z_1 = z_{11}$ and $z_2 = z_{21}^R$, where

$$
\begin{aligned}
z_{11} &= m((x_{11})(t_{11}t_{21})(x_{21}),(y_{11})(v_{21}w_{11}w_{21}y'_{21}))\\
z_{21} &= m((x_{31})(t_{31}t_{32})(x_{32}),(y_{31})(v_{32}w_{31}w_{32}y'_{32}))
\end{aligned}
$$

then also $z' = z'_1z'_2 \in L$, with $z'_1 = z'_{11}$ and $z'_2 = {z'_{21}}^R$, where

$$
\begin{aligned}
z'_{11} &= m((x_{11})(t_{11}t_{21})(x_{21}),(y_{11})(v_{21}y_{11}v_{21}w_{11}w_{21}y'_{21}y'_{21}))\\
z'_{21} &= m((x_{31})(t_{31}t_{32})(x_{32}),(y_{31})(v_{32}y_{31}v_{32}w_{31}w_{32}y'_{32}y'_{32}))
\end{aligned}
$$

from which it follows

$$z_{ij} = m(\overline{x}_{ij}, \overline{y}_{ij}\, v_{ij}\, w_{ij}\, \overline{u}_{ij})$$

and

$$z'_{ij} = m(\overline{x}_{ij}, \overline{y}_{ij} v_{ij} y_{ij}\, v_{ij}\, w_{ij}\, u_{ij}\, \overline{u}_{ij})$$

with the lists

$$
\begin{aligned}
&\overline{x}_{11} = (x_{11})(t_{11}t_{21})(x_{21}), \quad \overline{y}_{11} = (y_{11})(\varepsilon) \quad \text{and} \quad \overline{u}_{11} = (y'_{21})\\
&\overline{x}_{21} = (x_{31})(t_{31}t_{32})(x_{32}), \quad \overline{y}_{21} = (y_{31})(\varepsilon) \quad \text{and} \quad \overline{u}_{21} = (y'_{32})
\end{aligned}
$$

Case 3 In the path $[A_1^{(1)}, A_2^{(1)}]$ there exists a symbol belonging to $V_N^{(3)}$. That is:

$$
\begin{aligned}
S &\stackrel{+}{\Rightarrow}_G\ x_{11}(A_1^{(1)}\beta_{11})_1(\xi_{21})_2(\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}(\beta_{11})_1(\eta_{21}\xi_{21})_2(\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}(\beta_{21}\eta_{21}\xi_{21})_2(\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}(\eta_{21}\xi_{21})_2(b_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}y_{21}(x_{21})_2(\eta_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}(\xi_{32}\eta_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G\ x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}(\eta_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3
\end{aligned}
$$

At this point there exist in the third list component, or pd tape, two strings descending from the same node $A_1^{(1)}$, i.e. $\eta_{32}$ and $\eta_{31}$.

Case 3.1 Suppose $A_2^{(1)}$ is a descendant of $\eta_{32}$. Thus:

$$\begin{aligned}
S &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}(\eta_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}(B^{(3)}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}(A_2^{(1)})_1(\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}w_{11}(\alpha_{21})_2(\alpha_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}w_{11}w_{21}(\alpha_{32}\alpha_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}w_{11}w_{21}w_{32}(\alpha_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}w_{11}w_{21}w_{32}w_{31}(\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}w_{11}w_{21}w_{32}w_{31}y'_{32}t_{32}t_{31}y_{31}\xi_{31}
\end{aligned}$$

where we have derived $A_1^{(1)}$ as

$$\begin{aligned}
A_1^{(1)} &\stackrel{+}{\Rightarrow}_G y_{11}(\eta_{21})_2(\eta_{31})_3\\
&\stackrel{+}{\Rightarrow}_G y_{11}y_{21}(\eta_{32}\eta_{31})_3\\
&\stackrel{+}{\Rightarrow}_G y_{11}y_{21}v_{32}(B^{(3)}\eta'_{32}\eta_{31})_3\\
&\stackrel{+}{\Rightarrow}_G y_{11}y_{21}v_{32}(A_2^{(1)})_1(\eta'_{32}\eta_{31})_3\\
&\stackrel{+}{\Rightarrow}_G y_{11}y_{21}v_{32}w_{11}(\alpha_{21})_2(\alpha_{31}\eta'_{32}\eta_{31})_3\\
&\stackrel{+}{\Rightarrow}_G y_{11}y_{21}v_{32}w_{11}w_{21}(\alpha_{32}\alpha_{31}\eta'_{32}\eta_{31})_3\\
&\stackrel{+}{\Rightarrow}_G y_{11}y_{21}v_{32}w_{11}w_{21}w_{32}w_{31}y'_{32}y_{31}
\end{aligned}\tag{12}$$

and $A_2^{(1)}$ as

$$\begin{aligned}
A_2^{(1)} &\stackrel{+}{\Rightarrow}_{G'} w_{11}(\alpha_{21})_2(\alpha_{31})_3\\
&\stackrel{+}{\Rightarrow}_{G'} w_{11}w_{21}(\alpha_{32}\alpha_{31})_3\\
&\stackrel{+}{\Rightarrow}_{G'} w_{11}w_{21}w_{32}w_{31}
\end{aligned}$$

Hence, if we expand $A_2^{(1)}$ via the derivation (12), we have:

$$\begin{aligned}
S &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}(A_2^{(1)})_1(\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}(\eta_{21})_2(\eta_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}y_{21}(\eta_{32}\eta_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}y_{21}v_{32}(B^{(3)}\eta'_{32}\eta_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}y_{21}v_{32}(A_2^{(1)})_1(\eta'_{32}\eta_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}y_{21}v_{32}w_{11}(\alpha_{21})_2(\alpha_{31}\eta'_{32}\eta_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}y_{21}v_{32}w_{11}w_{21}(\alpha_{32}\alpha_{31}\eta'_{32}\eta_{31}\eta'_{32}\beta_{32}\beta_{31}\eta_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{32}v_{32}y_{11}y_{21}v_{32}w_{11}w_{21}w_{32}w_{31}y'_{32}y_{31}y'_{32}t_{32}t_{31}y_{31}x_{31}
\end{aligned}$$

i.e. both $z = z_1$ and $z = z_1' \in L$, where

$$z_1 = m\begin{pmatrix}(x_{11})(t_{11}t_{21})(x_{21}x_{31})(t_{32}t_{31})(x_{31})\\(y_{11})(y_{21})(v_{32}w_{11}w_{21}w_{31}y'_{21})(y_{31})\end{pmatrix}$$

and

$$z_1' = m\begin{pmatrix}(x_{11})(t_{11}t_{21})(x_{21}x_{31})(t_{32}t_{31})(x_{31})\\(y_{11})(y_{21})(v_{32}y_{11}y_{21}v_{32}w_{11}w_{21}w_{31}y'_{32}y_{31}y'_{32})(y_{31})\end{pmatrix}$$

where $m\begin{pmatrix}\overline{x}\\ \overline{y}\end{pmatrix}$ means $m(\overline{x},\overline{y})$, i.e. both $z$ and $z'$ have the form

$$z = m(\overline{x},\overline{y}v u w \overline{u}) \quad \text{and} \quad z' = m(\overline{x},\overline{y}v y v w u \overline{u})$$

with the lists

$$\begin{aligned}\overline{x} &= (x_{11})(t_{11}t_{21})(x_{21}x_{31})(t_{32}t_{31})(x_{31})\\ \overline{y} &= (y_{11})(y_{21})(\varepsilon)\\ \overline{u} &= (y'_{32})(y_{31})\end{aligned}$$

Case3.2 Suppose now $A_1^{(1)}$ is a descendant of $\eta_{31}$. Then:

$$\begin{aligned}S &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}(B^{(3)}\eta'_{31}\xi_{31})_3\\ &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}(A_2^{(1)})_1(\eta'_{31}\xi_{31})_3\\ &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}w_{11}(\alpha_{21})_2(\alpha_{31}\eta'_{31}\xi_{31})_3\\ &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}w_{11}w_{21}(\alpha_{32}\alpha_{31}\eta'_{31}\xi_{31})_3\\ &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}w_{11}w_{21}w_{32}w_{31}y'_{31}x_{31}\end{aligned}$$

where we have derived $A_1^{(1)}$ as

$$\begin{aligned}A_1^{(1)} &\stackrel{+}{\Rightarrow}_G y_{11}(\eta_{21})_2(\eta_{31})_3\\ &\stackrel{+}{\Rightarrow}_G y_{11}y_{21}(\eta_{32}\eta_{31})_3\\ &\stackrel{+}{\Rightarrow}_G y_{11}y_{21}y_{32}v_{31}(B^{(3)}\eta'_{31})_3\\ &\stackrel{+}{\Rightarrow}_G y_{11}y_{21}y_{32}v_{31}(A_2^{(1)})_1(\eta'_{31})_3\\ &\stackrel{+}{\Rightarrow}_G y_{11}y_{21}y_{32}v_{31}w_{11}(\alpha_{21})_2(\alpha_{31}\eta'_{31})_3\\ &\stackrel{+}{\Rightarrow}_G y_{11}y_{21}y_{32}v_{31}w_{11}w_{21}w_{32}w_{31}y'_{31}\end{aligned} \tag{13}$$

and $A_2^{(1)}$ as

$$\begin{aligned}A_2^{(1)} &\stackrel{+}{\Rightarrow}_G w_{11}(\alpha_{21})_2(\alpha_{31})_3\\ &\stackrel{+}{\Rightarrow}_G w_{11}w_{21}(\alpha_{31})_3\\ &\stackrel{+}{\Rightarrow}_G w_{11}w_{21}w_{32}w_{31}\end{aligned}$$

Hence again, if we use the derivation (13) to expand $A_2^{(1)}$, we get:

$$
\begin{aligned}
S &\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}(\eta_{21})_2(\eta_{31}\eta'_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}y_{21}(\eta_{32}\eta_{31}\eta'_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}y_{21}y_{32}(\eta_{31}\eta'_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}y_{21}y_{32}v_{31}(B^{(3)}\eta'_{31}\eta'_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}y_{21}y_{32}v_{31}(A_2^{(1)})_1(\eta'_{31}\eta'_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}y_{21}y_{32}v_{31}w_{11}(\alpha_{32})_2(\alpha_{31}\eta'_{31}\eta'_{31}\xi_{31})_3\\
&\stackrel{+}{\Rightarrow}_G x_{11}y_{11}t_{11}t_{21}y_{21}x_{21}x_{31}y_{32}t_{32}t_{31}v_{31}y_{11}y_{21}y_{32}v_{31}w_{11}w_{21}w_{32}w_{31}y'_{31}y'_{31}x_{31}
\end{aligned}
$$

so that both strings $z = z_1$ and $z = z'_1$ belong to $L$, where

$$z_1 = m\begin{pmatrix}(x_{11})(t_{11}t_{21})(x_{21}x_{31})(t_{32}t_{31})(x_{31})\\(y_{11})(y_{21})(y_{32})(v_{31}w_{11}w_{21}w_{32}w_{31}y'_{31})\end{pmatrix}$$

and

$$z'_1 = m\begin{pmatrix}(x_{11})(t_{11}t_{21})(x_{21}x_{31})(t_{32}t_{31})(x_{31})\\(y_{11})(y_{21})(y_{32})(v_{31}y_{11}y_{21}y_{32}v_{31}w_{11}w_{21}w_{32}w_{31}y'_{31}y'_{31})\end{pmatrix}$$

where $m\left(\frac{\overline{x}}{\overline{y}}\right)$ means $m(\overline{x},\overline{y})$, i.e. both $z$ and $z'$ have the form

$$z = m(\overline{x},\overline{y}uw\overline{u}) \quad \text{and} \quad z' = m(\overline{x},\overline{y}vyvwu\overline{u})$$

with the lists

$$
\begin{aligned}
\overline{x} &= (x_{11})(t_{11}t_{21})(x_{21}x_{31})(t_{32}t_{31})(x_{31})\\
\overline{y} &= (y_{11})(y_{21})(y_{32})(\varepsilon)\\
\overline{u} &= (y'_{31})
\end{aligned}
$$

The general case ($n > 3$) can be deduced in a similar way.