Cancellation in context-free languages: enrichment by reduction

M. Jantzen* H. Petersen‡

December 10, 1996

Abstract

The following problem is shown to be decidable: Given a context-free grammar G and a string $w \in X^*$, does there exist a string $u \in L(G)$ such that w is obtained from u by deleting all substrings u_i that are elements of the symmetric Dyck set D_1^* ?

The intersection of any two context-free languages can be obtained from only one context-free language by cancellation either with the smaller semi-Dyck set $D_1'^* \subset D_1^*$ or with D_1^* itself.

Also, the following is shown here for the first time: If the set $EQ := \{x^n\bar{x}^n \mid n \in \mathbb{N}\} \subset D_1'^*$ is used for this cancellation, then each recursively enumerable set can be obtained from linear context-free languages.

Previous work has shown that cancellation of substrings from the semi-Dyck language $D_1'^*$ or from any of the former languages D_1^*, or EQ, allows to obtain the following from the context-free languages: any terminal Petri net language, [14], [16], the intersections of any context-free with a terminal Petri net language, and the nested iterated substitution thereof, [15]. We generalize this characterization by showing that linear context-free languages suffice for generating terminal Petri net languages. In proving this we obtain a new closure property of the family of Petri net languages which is not shared by the context-free sets.

Introduction

Adding a further control mechanism to the somewhat limited power of context-free grammars is a rich source of new insights into the field of formal language theory. Let us only mention matrix grammars, indexed grammars and programmed grammars (see [5]). In this paper we will investigate another approach: a context-free language is modified by a reduction $x\bar{x} \to \lambda$. In fact, the iterated application of this reduction reduces the well known semi-Dyck set $D_1'^*$ over one pair of parentheses to the empty word λ. Likewise, the two-sided reduction $x\bar{x} \to \lambda$, $\bar{x}x \to \lambda$ yields the symmetric Dyck set D_1^*.

Dyck reductions were studied in [1, 2, 3, 4, 6, 10, 13, 23, 24, 25] mostly by considering general Dyck sets having more than one type of parentheses for the cancellation which then

*Fachbereich Informatik, Universität Hamburg, e-mail: jantzen@informatik.uni-hamburg.de

‡since Feb.1995: Institut für Informatik, Univ. Stuttgart, e-mail: petersen@informatik.uni-stuttgart.de

yields the full class of recursively enumerable sets. This result is very famous and can be found in several places, see [10, 25, 22]. It has been shown in [6], that even linear context-free languages are sufficient, a result recently rediscovered in [3]. In the case of only one pair of parentheses it is not yet clear whether the generated class of languages is equally rich. The authors extended the results of Savitch on cancellation grammars in [14, 15] and showed that the cancellation of complete substrings from the semi-Dyck language $D_1'^*$ in context-free languages not only yielded the full class $\underline{L}$ of terminal Petri net languages, see [11, 12, 21], but moreover their elementwise intersection with the context-free languages, i.e. the family $\underline{L} \wedge \underline{Cf}$. The simulations used are even strong enough for generating the closure with respect to nested iterated substitution and non-erasing homomorphisms of the above language class.

Kimura uses the cancellation of two symbols as a means for describing restricted communication between systems [18]. The context-free language models the system, the gaps between non parentheses symbols represent communication channels and the parentheses model signal resp. wait messages. This point of view resembles coupled producer-consumer systems and therefore the relation to Petri net languages seems natural. Another related area are the reductions of picture words, where movements of some device — like a plotter pen, electron ray or laser — on a grid are minimized [4, 20]. Retreats, i.e. movements of the device in one and then in the opposite direction, are deleted from regular and context-free languages. Obviously we deal with the one dimensional case of those problems.

Lila Kari's thesis [17] investigates a variety of different operations on formal languages. In that work she covers what she calls "iterated deletion", an operation similar to the one used here. Her overview over possible approaches to deletion and insertion in formal languages justifies an in-depth treatment like ours.

Since it is known [1, 2, 13], that regular sets are closed with respect to Dyck-reductions, the question arose which class of languages would be obtained by applying the cancellation with the semi-Dyck set $D_1'^*$ to the slightly larger class of linear context-free sets. As a first step in this direction we show here, see also [16], how the class of terminal Petri net languages can be obtained from the linear context-free languages, (abb. $\underline{Lin}$), by complete cancellation of substrings from any of the sets EQ, $D_1'^*$, or D_1^*.

To obtain that result the vectors $m \in \mathbb{N}^S$ are encoded by a string of opening brackets x of length $c(m)$. The encoding homomorphism $c : \mathbb{N}^S \to \mathbb{N} \setminus \{0\}$ uses a prime decomposition:

$$c\big((m_1, m_2, \ldots m_k, \ldots m_{|S|})\big) := p_1^{m_1} p_2^{m_2} \cdots p_k^{m_k} \cdots p_{|S|}^{m_{|S|}}$$

where p_k denotes the k^{th} prime number in the sequence $2, 3, 5, \ldots$.

The new result presented here will show, that even the intersections of two context-free languages form a subclass of $\underline{Cf}\%D_1^*$ and moreover their closure with respect to non-erasing homomorphisms and nested iterated substitution. To represent the intersection of any two context-free languages by cancellation from only one such language a different n-ary encoding will be used. The comparison of the proofs reveals how reductions can be used in simulations of different storage devices like counters and pushdown stacks, and so our recent results not only add to the evolving knowledge about this interesting type of languages but might help in thoroughly understanding the power of reductions.

Definitions and Results

Definition 0.1 *Let $\underline{Reg}$ (resp. $\underline{Lin}$, $\underline{Cf}$, $\underline{Cf^{\triangle}}$, $\underline{Rec}$, $\underline{RE}$) denote the family of regular sets (linear context-free, context-free, λ-free context-free, recursive, and recursively enumerable sets, respectively).* □

Definition 0.2 *Let $X := \{x_i, \bar{x}_i \mid 1 \leq i \leq k\}$ be an alphabet of k pairs of matching parentheses, where x_i is the opening and $\bar{x}_i$ the closing bracket. Cancelling a matching pair $x_i\bar{x}_i$ $(1 \leq i < k)$, within a string $ux_i\bar{x}_iv \in V^*$, where V is any alphabet containing X_k , yields the word uv. This one-step reduction is denoted by $\triangle_k(ux_i\bar{x}_iv) := uv$. $D_k'^*$ denotes the one-sided (or semi-) Dyck set of all strings $w \in X_k^*$ which can be reduced to the empty string λ by finitely many cancellations of type $\triangle_k$. Instead of using the reduction based on $\triangle_k$ as in [2, 15] we follow Kimura[18] and use the cancellation of subwords on a language $K \subseteq V_1^*$ with any set $L \subseteq V_2^*$ defined by:*

$$
\begin{aligned}
&K\%L := \\
&\quad \{w \in (V_1 \setminus V_2)^* \mid \exists w_1 \in K\ \exists n \in \mathbb{N}\ \forall 0 \leq i \leq n\ \exists u_i \in V_1^*\ \exists v_i \in L : \\
&\quad w_1 = u_0v_1u_1 \ldots v_nu_n \text{ and } w = u_0u_1 \ldots u_n\}
\end{aligned}
$$

$K\%L$ is called a Dyck$_k$-reduction if L equals the Dyck set $D_k'^*$. Whenever we use the Dyck sets D_1^*, $D_1'^*$, or $EQ := \{x^n\bar{x}^n \mid n \in \mathbb{N}\}$ we use the brackets x and $\bar{x}$ from the alphabet $X := \{x, \bar{x}\}$ and omit the index 1 in X_k that is used only in the sets $D_k'^*$ where $k \geq 2$.

The canonical extension to families $\mathcal{K}$ and $\mathcal{L}$ is defined by

$$\mathcal{K}\%\mathcal{L} := \{K\%L \mid K \in \mathcal{K} \text{ and } L \in \mathcal{L}\}$$

and we will write $\mathcal{K}\%L$ as a shorthand for $\mathcal{K}\%\{L\}$.

For a class of languages $\mathcal{C}$ we denote by $H(\mathcal{C})$ the closure of $\mathcal{C}$ with respect to non-erasing homomorphisms. If τ is a substitution such that $a \in \tau(a)$ for any symbol a, then we denote by $\tau^{\infty}(L)$ the operation of nested iterated substitution of the language L, the union of all languages obtained by finitely often applying τ to L. A more detailed definition of this operation may be found in [9] . For a language class $\mathcal{C}$ we define the closure of $\mathcal{C}$ with respect to nested iterated substitution $\tau^{\infty}(\mathcal{C})$ to be the smallest language class containing $\mathcal{C}$ and being closed under nested iterated substitution.

For any family $\mathcal{C}$ let $\mathcal{C}^{\triangle} := \{L \setminus \{\lambda\} \mid L \in \mathcal{C}\}$ be a class of λ-free languages. Often, but not always we find $\mathcal{C}^{\triangle} \subseteq \mathcal{C}$.

□

It has been shown in [15] that the emptiness problem for the family $\underline{Cf}\%D_1'^*$ is undecidable. That proof at the same time shows that undecidability result for the families $\underline{Cf}\%D_1^*$ and $\underline{Cf}\%EQ$, because only subwords from the set $EQ \subseteq D_1'^* \subseteq D_1^*$ were used for cancellation. The problem, as to whether the family $\underline{Cf}\%D_1'^*$ only contains recursive sets is approached, though not solved, by the following two results:

Theorem 1 *$\mathcal{L}\%D_1^* \subseteq \underline{Rec}$, for any family $\mathcal{L}$ of languages closed w.r.t. inverse homomorphisms, intersection with regular sets and for which we can effectively compute the Parikh image of every language.*

Theorem 2 $\underline{Lin}\%EQ = \underline{RE}$

Proof of Theorem 1: The proof of the first result uses the decidability of the emptiness of semilinear vector sets, is fairly straight forward, and is only sketched. For any language $K \in \mathcal{L}$, $K \subseteq (V \cup X)^*$ and each string $w = w_1 w_2 \dots w_n$, where $w_i \in V \setminus X$, an algorithm will be given which decides, whether $w \in K\%D_1^*$.

For each i, $1 \leq i \leq n+1$, define $Y_i := \{x_i, \bar{x}_i\}$ and $h : Y_n \to X$ by $h(x_i) := x$ and $h(\bar{x}_i) := (\bar{x})$. Then

$$L_w := h^{-1}(K) \cap Y_1^*\{w_1\}Y_2^*\{w_2\}Y_3^* \cdots Y_n^*\{w_n\}Y_{n+1}^*$$

is a set of which one can decide emptiness. Now, $w \in K\%D_1^*$ only if $L_w \neq \emptyset$, and we have to decide whether there exists a string

$$u_1 w_1 u_2 w_2 u_3 \dots u_n w_n u_{n+1} \in L_w$$

such that $u_i \in Y_i^*$ is indeed an element of the symmetric Dyck set and has the same number of opening and closing brackets. To do this we look at the Parikh image $P := \psi(L_w) \subseteq \mathbb{N}^{2n+|V|}$ which forms a semilinear vector set. If $Q \subseteq \mathbb{N}^{2n+|V|}$ denotes the semilinear set which satisfies $m(x_i) = m(\bar{x}_i)$ for each $1 \leq i \leq n$ and each element $m \in Q$ and is arbitrary in the remaining coordinates, then $Q \cap P \neq \emptyset$ if and only if $w \in K\%D_1^*$. $\square$

Corollary 1 $\underline{Cf}\%D_1^* \subset \underline{Rec}$

Obviously the family $\underline{Cf}$ fulfills the desired closure properties, moreover a simple diagonalization by enumerating strings and context-free grammars shows proper containment: the language $\{w_i \mid w_i \notin L(G_i)\%D_1^*\}$ is recursive but not equal to $L(G_k)\%D_1^*$ for any index k. $\square$

The proof of Theorem 2 is more involved and the result should be compared with similar results by Frougny et al. [6] and Brandenburg, Dassow [3] who showed $\underline{Lin}\%D_2^* = \underline{RE}$.

Proof of Theorem 2: Let $E \subseteq (V \setminus X)^*$ be any recursively enumerable set which is accepted by $M = (\Sigma, Z, Z_{\text{start}}, Z_{\text{end}}, K)$, a nondeterministic one-way two counter automaton. Here $\Sigma := V \setminus X$ is the input alphabet, Z is the finite set of states with subsets Z_{start} and Z_{end} of initial and final states, respectively.

The two counters are named A (resp. B) and can each be altered by operations c_A (resp. c_B), where c_A and c_b are elements of $\{+1, -1, \text{test}\}$ with the following meaning:

$$c_A, c_B := \begin{cases} +1 & \text{add 1 to the respective counter} \\ -1 & \text{subtract 1 from the respective counter} \\ \textit{test} & \text{this move only if the respective counter stores zero} \end{cases}$$

$K \subseteq Z \times (\Sigma \cup \lambda) \times \{+1, -1, \text{test}\} \times \{+1, -1, \text{test}\} \times Z$ is the set of transitions of M and an element $(z_i, y, c_A, c_B, z_j) \in K$ has the obvious meaning: move from state z_i into state z_j, read the element y from the one-way input tape, and perform the operation c_A or c_B on the respective counter. M begins with an initial state and accepts $w \in \Sigma^*$ if the transitions lead M to a final state while reading w and both counters are empty. The contents of the two counters, $(m_1, m_2) \in \mathbb{N}^2$, is encoded unary by $p := 2^{m_1} \cdot 3^{m_2} \in \mathbb{N}$ and will be encoded by strings of the form x^p or $\bar{x}^p$. As in [16] , the change of counter A by c_A will be done by multiplying p by 2, if c_A is $+1$, by dividing by 2, if c_A is -1, or by checking whether division leaves a remainder of 1, if c_A is 'test'. Division will be done by successively subtracting 2 from p, which can be encoded by using 2 closing brackets $\bar{x}$ as often as possible. If counter B is used, the same is done with the prime 3 instead of 2.

In order to design a linear context-free grammar G_M we will need the following nonterminals:

$$\Sigma_Z := \{H_i \mid z_i \in Z\}$$

$$\Sigma_A := \{A_k \mid k \in K\}$$

$$\Sigma_B := \{B_k \mid k \in K\}$$

and S as the starting symbol of G_M.

The rules of G_M are collected in the following sets:

$\boldsymbol{\{S \to H_j\bar{x} \mid z_j \in Z_{\text{start}}\}}$ start M in an initial state with empty counters.

For each transition $k := (z_i, y, c_A, c_B, z_j) \in K$ the following sets of rules are defined :

$\boldsymbol{\{H_i \to yA_k\}}$ M reads y and starts to try the operation c_A on counter A, which means that the encoding of p by the string $\bar{x}^p$ appears to the right of the nonterminal A_k.

$\boldsymbol{\{A_k \to xA_kx^2 \mid \textbf{if } c_A = -1\}}$ the successive subtraction is done by opening 2 brackets until there are as many as there are closing brackets, which will finally be checked by cancelling with the set EQ.

$\boldsymbol{\{A_k \to x^2A_kx \mid \textbf{if } c_A = +1\}}$ twice as many brackets x are opened at the left of A_k as there appear closing brackets at the right.

$\boldsymbol{\{A_k \to x^2A_kx^2 \mid \textbf{if } c = \textbf{test}\}}$ the number p at the right of A_k is divided by 2, hopefully with a remainder and recovered at the left of A_k.

$\boldsymbol{\{A_k \to B_kx\bar{x} \mid \textbf{if } c_A = +1 \textbf{ or } c_A = -1\}}$ the additional pair $x\bar{x}$ leads to a string in EQ only if the string immediately to the right of these two symbols is already an element of EQ^*, remember: $L\%EQ = L\%EQ^*$. The symbol B_k means that, still with transition number k, counter B will now be altered according c_B.

$\boldsymbol{\{A_k \to xB_kx\bar{x}x \mid \textbf{if } c_A = \textbf{test}\}}$ the remainder 1 is subtracted, $x\bar{x}$ marks the empty counter A and the former counter content is correctly restored at the left hand side of the new symbol B_k.

$\boldsymbol{\{B_k \to \bar{x}^3B_k\bar{x} \mid \textbf{if } c_B = -1\}}$

$\{B_k \to \bar{x}B_k\bar{x}^3 \mid \text{if } c_B = +1\}$

$\{B_k \to \bar{x}^3 B_k\bar{x}^3 \mid \text{if } c_B = \text{test}\}$

$\{B_k \to x\bar{x}H_j \mid \text{if } c_B = +1 \text{ or } c_B = -1\}$

$\{B_k \to xx\bar{x}H_j\bar{x} \mid \text{if } c_B = \text{test}\}$

$\{B_k \to \bar{x}^2 x\bar{x}H_j\bar{x}^2 \mid \text{if } c_B = \text{test}\}$ these rules yield the altering of counter B by similar rules as before. If the next state z_j is a final state, and both counters are empty, then we can accept the string produced so far by the symbols in $V \setminus X$ and the nonterminal H_j may disappear through the last rule in the set:

$\{H_j \to x\bar{x} \mid z_j \in Z_{\text{end}}\}$ □

$L(G_M)\%EQ = E$ is thus proved, since G_M is linear context-free, the cancellation with the set EQ checks the contents of the counters and in addition tests that the moves of the counter automaton were performed properly.

Corollary 2

$$\begin{array}{rcl} \underline{Cf}\%D_1'^* & \subseteq & \underline{Lin}\%EQ \\ \underline{Cf}\%D_1^* & \subset & \underline{Lin}\%EQ \\ \underline{Lin}\%D_1'^* & \subset & \underline{Lin}\%EQ \end{array}$$

Proof: The inclusions follow directly from Theorem 2. Theorem 1 proves the second inclusion to be proper, and the inclusion $\underline{Lin}\%D_1'^* \subset \underline{Rec}$ has been shown in [15]. □

Definition 0.3 *A Place/Transition net (P/T-net, for short) $N := (S, T, W)$ consists of a set of places $S := \{s_1, s_2, \ldots, s_{|S|}\}$, a set $T := \{t_1, t_2, \ldots, t_{|T|}\}$ of transitions, and a weighting function $W : S \times T \cup T \times S \to I\!N$ which specifies $W(x, y)$ directed arcs from node x to node y. The vector t^- denotes the vector $W(\cdot, t) \in I\!N^S$, i.e. the smallest marking enabling t, and likewise t^+ denotes the vector $W(t, \cdot) \in I\!N^S$, respectively. A capacity function is not used here and the sets S and T are disjoint nonempty finite sets. Thus N is nothing but a finite bipartite graph.*

A Petri net system $\Sigma := (N, M_0, M_f, h, V)$ is specified by a P/T-net N, finite sets $M_0 \subseteq I\!N^S$ (resp. $M_f \subseteq I\!N^S$) of initial and final markings, respectively. V is an alphabet of labels for the transitions and $h : T \to V^+$ is a non-erasing labelling homomorphism which shall be omitted if it is an isomorphism between T and V.

Let $\Sigma := (N, M_0, M_f, h, V)$ be a Petri net system. Then we define the relation $[w\rangle$ on $I\!N^S \times I\!N^S$ for each $w \in T^*$ by:

a) $M[\lambda\rangle M$ for each marking $M \in I\!N^S$

b) $M[t\rangle M'$ if $t^- \geq M$ and $M' = M - t^- + t^+$ for each $t \in T$ and markings $M, M' \in I\!N^S$

c) $M[w\rangle M''$ if $w = ut$ for some $u \in T^*$, $t \in T$ and there exists M' such that $M[u\rangle M'$ and $M'[t\rangle M''$.

The terminal ($\underline{L}$-type in [21]) language of the system Σ is defined as follows:

$$L(\Sigma) := \{h(w) \mid \exists m_0 \in M_0, \exists m_f \in M_f : m_0[w\rangle m_f\}$$

We call any string $w \in T^*$ such that $m_0[w\rangle m$ holds a transition sequence, thereby distinguishing it from the so-called firing sequence $h(w)$. It was shown already by Hack in [11, Theorems 2.3, 5.8] that the class $\underline{L}$ of Petri net languages

$$\underline{L} := \{L(\Sigma) \mid \ \Sigma \text{ any Petri net system}\}$$

is identical to the set of those Petri net languages $L(\Sigma)$ where Σ satisfies the following normal form restrictions:

- *each of the sets M_0 and M_f contains only one single, non-zero marking m_0 (resp. m_f) i.e. , $m_0 \neq \underline{0} \neq m_f$. Only $m_f \neq \underline{0}$ amounts to some minor and obvious modification in [11].*
- *the labelling homomorphism is a coding, i.e. , $h : T \to V$.*
- *each transition of N in $\Sigma = (N, m_0, m_f, h, V)$ has at least one input and one output place, i.e. , $\forall t \in T\ \exists p, q \in S : W(p,t) \neq 0 \neq W(t,q)$.*
- *the underlying Petri net N has no selfloops, i.e. , $\forall p \in S : t^+(p) \cdot t^-(p) = 0$.*

$\square$

In [14] it was shown for the first time that terminal Petri net languages can be obtained by Dyck$_1$-reductions from the context free-languages, i.e. , $\underline{L} \subseteq \underline{Cf}\% D_1'^*$. This result is generalized here by Theorem 3:

Theorem 3 *$\underline{L} \subseteq \underline{Lin}\% D_1^*$, i.e. for each terminal Petri net language $K \in \underline{L}$ there exists a linear context-free language $L \in \underline{Lin}$ such that $K = L\% D_1^*$.*

The encoding of the reachable markings in a Petri net by strings of opening parentheses x will be the same as described in [14, 15] but the (labels of the) transitions can no longer be generated by the underlying linear grammar in the original sequence of their occurrence in the firing sequence.

Thus, in order to prove Theorem 3 we first have to show that the family $\underline{L}$ of Petri net languages is closed with respect to a new operator based on the operation of twisting the firing sequences as defined as follows:

Definition 0.4 *The operation twist* : $V^* \to V^*$ *is defined recursively for any* $w \in V^*$*, and symbols* $a, b \in V$*, by:*

$$\begin{aligned} twist(awb) &:= ab \cdot twist(w) \\ twist(a) &:= a \\ twist(\lambda) &:= \lambda \end{aligned}$$

The operator TWIST is defined on families of languages $\mathcal{L}$ *in the usual way by:*

$$TWIST(\mathcal{L}) := \{twist(L) \mid L \in \mathcal{L}\}.$$

It is not hard to see that the family of context-free languages $\underline{Cf}$ is *not* closed under TWIST: Consider the language $K := \{a^{2n}b^m c^m d^n \mid n, m \in I\!N\} \in \underline{Lin} \subseteq \underline{Cf}$, then twist$(K)$ is not context-free, since

$$\text{twist}(K) \cap \{ad\}^*\{ac\}^*\{b\}^* = \{(ad)^n(ac)^n b^n \mid n \geq 0\}$$

Note also, that with COPY $:= \{ww \mid w \in V^*\}$ and PAL $:= \{ww^{\text{rev}} \mid w \in V^*\}$ one has twist(COPY) = PAL, and twist(PAL) $= \{yy \mid y \in V\}^* \in \underline{Reg}$.

It is not surprising that regular sets are closed under twist. We will show this here, even though it could serve as an exercise, since the main idea used is essentially the same that we will apply to the Petri nets in proving Theorem 5. There, however, the construction is not as simple as used in the following proof.

Theorem 4 *TWIST*$(\underline{Reg}) \subseteq \underline{Reg}$ *i.e. , twist*(R) *is regular for any regular set* $R \in \underline{Reg}$.

Proof: Let $A = (V, Z, K, Z_{\text{start}}, Z_{\text{end}})$ be an arbitrary finite automaton with input alphabet V, Z the set of states, Z_{start} and Z_{end} being the subsets of initial and final states, respectively, and $K \subseteq Z \times V \times Z$ the set of moves.

Then define $A^{\text{rev}} := (V, \hat{Z}, \hat{K}, \hat{Z}_{\text{start}}, \hat{Z}_{\text{end}})$ by $\hat{Z} := \{\hat{z} \mid z \in Z\}$, $\hat{Z}_{\text{start}} := Z_{\text{end}}$, $\hat{Z}_{\text{end}} := Z_{\text{start}}$, and $\hat{K} := \{(\hat{z}_2, x, \hat{z}_1) \mid (z_1, x, z_2) \in K\}$. It should be clear that $(L(A))^{\text{rev}} = L(A^{\text{rev}})$. Based on A and A^{rev} a variant of the product automaton is defined by: $B := (V, Q, K^\bullet, Q_{\text{start}}, Q_{\text{end}})$ with state sets $Q := Z \times \hat{Z} \times \{0, 1\}$, $Q_{\text{start}} := Z_{\text{start}} \times \hat{Z}_{\text{start}} \times \{0\}$, and $Q_{\text{end}} := \{[z, \hat{z}, 0], [z, \hat{z}, 1] \mid z \in Z\}$. $K^\bullet$ is defined by:

$$\forall z_3 \in Z : ([z_1, \hat{z}_3, 0], x, [z_2, \hat{z}_3, 1]) \in K^\bullet \text{ iff } (z_1, x, z_2) \in K$$

$$\forall z_1 \in Z : ([z_1, \hat{z}_3, 1], x, [z_1, \hat{z}_4, 0]) \in K^\bullet \text{ iff } (\hat{z}_3, x, \hat{z}_4) \in \hat{K}.$$

We claim that $L(B) = \text{twist}(L(A))$. Automaton B works on an input

$$w := x_1x_2x_3x_4x_5x_6x_7 \dots x_{n-2}x_{n-1}x_n$$

as follows: the sequence $\alpha := x_1x_3x_5x_7 \cdots$ forms a prefix of a string accepted by A whereas $\beta := x_2x_4x_6 \cdots$ forms a prefix of some string accepted by A^{rev}, hence β^{rev} is a suffix of some string accepted by A. The path accepting the prefix will be followed in the first components of the states in Q and that for the suffix in the second component. The third component indicates by 0 (resp. by 1) that automaton A (automaton A^{rev}, resp.) is simulated. If an accepting state $[z, \hat{z}, p]$ for $p \in \{0,1\}$ is reached in B on input w, where $p = 1$ if the length of w is even and $p = 0$ otherwise, then the state z is reached in A from an initial state by input of the prefix α and from this state z the word β^{rev} leads A into an accepting state. This means that A can accept the string $v := \alpha\beta^{\text{rev}} = x_1 x_n x_3 x_{n-1} x_5 x_{n-2} x_7 \cdots x_6 x_4 x_2$ for which one has $\text{twist}(v) = w$. □

Theorem 5 *$TWIST(\underline{L}) \subseteq \underline{L}$, i.e. , twist$(L)$ is a terminal Petri net language for any Petri net language $L \in \underline{L}$.*

Proof: The proof proceeds in several stages, of which the first two are trivial. First we consider only unlabelled Petri net systems $\Sigma := (N, m_0, m_f, h, V)$, i.e. those where the homomorphism h is the identity, and each firing sequence coincides with its transition sequence and constructively prove that $\text{twist}(L(\Sigma))$ is again a transition sequence of some unlabelled Petri net system $\vec{\Sigma}$. Any labelling homomorphism h introduced later, by the normal form restriction is a coding such that $\text{twist}(h(L)) = h(L(\vec{\Sigma}))$ is immediate and the result will follow in its full generality. Secondly, for any given (unlabelled) Petri net system $\Sigma := (N, m_0, m_f, V)$ with $N := (S, T, W)$, and $L := L(\Sigma)$ define a copy $\bar{N} := (\bar{S}, \bar{T}, \bar{W})$ of N with all arcs reversed as follows: $\bar{S} := \{\bar{s} \mid s \in S\}$, $\bar{T} := \{\bar{t} \mid t \in T\}$, and $\forall \bar{s} \in \bar{S}\ \forall \bar{t} \in \bar{T} : (\bar{W}(\bar{s}, \bar{t}) := W(t, s)$ and $\bar{W}(\bar{t}, \bar{s}) := W(s, t))$. Based on $\bar{N}$ a new Petri net system $\bar{\Sigma} := (\bar{N}, \bar{m}_0, \bar{m}_f, V)$ is given by setting $\bar{m}_0 := m_f$, $\bar{m}_f := m_0$. It is easily verified that $L(\bar{\Sigma}) = (L(\Sigma))^{\text{rev}}$. If in later steps a labelling homomorphism h is defined for Σ it will be extended to $\bar{\Sigma}$ by setting $h(\bar{t}) := h(t)$ for each $\bar{t} \in \bar{T}$ and then obviously $h(L(\bar{\Sigma})) = (h(L(\Sigma))^{\text{rev}}$.

Having done this, we construct a Petri net system $\vec{\Sigma} := (\vec{N}, \vec{m}_0, \vec{M}_f, \vec{h}, V)$ that combines Σ and $\bar{\Sigma}$ into a single system $\vec{N} := (\vec{S}, \vec{T}, \vec{W})$ which simulates both nets N and $\bar{N}$ independently of each other, starting with N and then alternating them. This new system is obtained by adding two toggling places p and q with one token to the disjoint union of N and $\bar{N}$ in such a way that a transition of N in $\vec{\Sigma}$ can occur only if p is marked and in such a case moves the only token to the place q. If q is marked, then only the transitions of $\bar{N}$ can occur and by its firing move the token back to the place p. The initial marking $\vec{m}_0$ of $\vec{\Sigma}$ will be defined to be equal to those of the subnets N and $\bar{N}$ on the corresponding places, i.e. , $\forall s \in S : \vec{m}_0(s) := m_0(s)$, $\forall \bar{s} \in \bar{S} : \vec{m}_0(\bar{s}) := \bar{m}_0(\bar{s})$, $\vec{m}_0(p) := 1$, and $\vec{m}_0(q) := 0$. The set of final markings $\vec{M}_f$ will be defined later since the construction is not yet completed.

Obviously, we have to define $\vec{S} := S \cup \bar{S} \cup \{p, q\}$, $\vec{T} := T' \cup \bar{T}'$, where transitions in T' (resp. in $\bar{T}'$) behave exactly as those in T (resp. in $\bar{T}$) with the only modification that in addition they move the token from p to q, while the transitions in $\bar{T}'$ move it back to place p. The new arc weighting matrix $\vec{W}$ will be defined later on the base of the old W and $\bar{W}$ (to which sets we then will refer). The sets T' and $\bar{T}'$ will be enlarged even further in the following. At this stage of the construction the system $\vec{\Sigma}$ will never reach the zero marking $\underline{0}$ by firing transitions because either place p or place q is marked. Moreover, $m \in [\vec{m}_0\rangle$ implies $\sum_{s \in \vec{N}} m(s) \geq 3$, since the total number of tokens in N (and hence also in $\bar{N}$ and in $m(p) + m(q)$ for each reachable marking m) never dips below 1. This is due to the normal form requirement in Definition 3.

The transition sequences $w \in \vec{T}$ with $\vec{m}_0[w\rangle\vec{m}_f$ in $\vec{\Sigma}$ all have a decomposition of the following form:

$$w = t'_{i_1}\bar{t}'_{j_n}t'_{i_2}\bar{t}'_{j_{n-1}}t'_{i_3}\bar{t}'_{j_{n-2}}\cdots t'_{i_{n-k}}\bar{t}'_{j_{k+1}}\cdots t'_{i_{n-1}}\bar{t}'_{j_2}t'_{i_n}\bar{t}'_{j_1}$$

where $t_{i_1}t_{i_2}\cdots t_{i_n} \in L(\Sigma)$ and $\bar{t}_{j_n}\bar{t}_{j_{n-1}}\cdots\bar{t}_{j_1} \in L(\bar{\Sigma})$ are transition sequences in the original nets. The primed transitions in $\vec{\Sigma}$ are those having the same label but having been modified regarding the toggling places. Obviously, $t_{j_1}\cdots t_{j_{n-1}}t_{j_n}$ is a transition sequence in N, but not necessarily the same as $t_{i_1}t_{i_2}\cdots t_{i_n}$ because N and $\bar{N}$ behave completely independent of each other.

Let us now assume, for the sake of an easier understanding, that each transition sequence w with $m_0[w\rangle m_f$ in N has even length. This can be done without care, since the construction can simply be modified to also cover transition sequences of odd length. Only the final markings have to be changed and this is done similarly as in the previous proof of Theorem 4 for the finite automata.

Now, if $|w| = n = 2r$ is even and both subnets N and $\bar{N}$ use the same transition sequence $v = \bar{v}^{\text{rev}}$, i.e. , $t_{i_k} = \bar{t}_{j_k}$ for all $1 \leq k \leq n$, then the marking reached from $\vec{m}_0$ after exactly $r = n/2$ steps is identical in the corresponding places of the subnets N and $\bar{N}$:

$$\vec{m}_0[t'_{i_1}\bar{t}'_{j_n}t'_{i_2}\bar{t}'_{j_{n-1}}t'_{i_3}\bar{t}'_{j_{n-2}}\cdots t'_r\bar{t}'_{r+1}\rangle m$$

implies $m(s) = m(\bar{s})$ for each $s \in S$, $m(p) = 1$, and $m(q) = 0$.

If, on the other hand, $u = u_1u_2\cdots u_r$ with $u_i \in T$ (resp. $v = v_1v_2\cdots v_r$, $v_j \in \bar{T}$) are transition sequences in N (resp. in $\bar{N}$) that reach the same marking in the corresponding places, then uv^{rev} is a transition sequence in N. Since $|u| = |v|$ we have the following transition sequence in $\vec{\Sigma}$ by using the modified transitions u'_i and v'_i: $\vec{m}_0[u'_1v'_1u'_2v'_2\cdots u'_rv'_{r+1}\rangle m$, such that $m(p) = 1$ and $m(s) = m(\bar{s})$.

Thus, in order to use the system $\vec{\Sigma}$ for generating the language

$$\begin{aligned}&\text{twist}(L(\vec{\Sigma})) = \\ &\quad \{w \mid w = u_1v_{2r}u_2v_{2r-1}\cdots u_rv_{r+1} \text{ with } u_1u_2\cdots u_rv_{r+1}v_{r+2}\cdots v_{2r} \in L(\Sigma)\}\end{aligned}$$

we would need the following infinite set of final markings

$$M_f := \{m \in \mathbb{N}^{\vec{S}} \mid m(p) = 1, m(q) = 0, \text{ and } \forall s \in S : m(s) = m(\bar{s})\}$$

The first idea to use λ-transitions as silent moves for checking the equality of the corresponding places in the final markings to be reached must be taken with great care. It is known that λ-transitions increase the generating power of Petri nets, see for instance [12]. We can do this in some controlled way by performing one silent move which decreases s by $\bar{W}(\bar{t}, \bar{s})$ (resp. $\bar{s}$ by $W(t, s)$) after transition t of the subnet N (resp. of $\bar{N}$) has added one token to the place s (resp. $\bar{s}$). Hence, if transition t has to add a token to place s it might instead be tested whether the corresponding token is already present on the corresponding place $\bar{s}$ by removing it.

In the example below transition t' does this testing and its effect is the same as that of transition t followed by a silent λ-transiton which has s and $\bar{s}$ as its only input places.

Figure 1: A simple example.

Following this idea we will add to $\vec{\Sigma}$ all possibilities of new transitions that combine an ordinary transition from T or $\bar{T}$ with a λ-transition in the indicated way. Since the marking reached in r steps from m_0 cannot be larger than $r \cdot k + m_0(s)$ for k being the maximum of tokens a transition can ever put on s in a single step, we need at most r such 'λ-transitions' in order to reach a marking in which only one of the toggle places is marked.

The final markings to be reached then should be zero on all but one of places p (or q, if the transition sequence has odd length). Therefore, the set of final markings $\vec{M}_f := \{\vec{m}_{\text{even}}, \vec{m}_{\text{odd}}\} \subset \mathbb{N}^{\vec{S}}$ is defined by:

$$\vec{m}_{\text{even}}(y) := \begin{cases} 1 & \text{if } y = p \\ 0 & \text{otherwise} \end{cases}$$

$$\vec{m}_{\text{odd}}(y) := \begin{cases} 1 & \text{if } y = q \\ 0 & \text{otherwise} \end{cases}$$

For each transition $t \in T$ define a new set:

$$\begin{aligned} \Gamma(t) := \{t_z \mid z \leq t^+ \text{ with } \quad & W(s, t_z) := W(s, t) \text{ and} \\ & W(\bar{s}, t_z) := W(t, s) - z(s) = (t^+ - z)(s) \text{ and} \\ & W(t_z, s) := z(s) \text{ and } W(t_z, \bar{s}) := 0 \text{ and} \\ & W(t_z, p) := W(q, t_z) := 0 \text{ and} \\ & W(p, t_z) := W(t_z, q) := 1 \text{ for each } s \in S\} \end{aligned}$$

Likewise, for each transition $\bar{t} \in \bar{T}$ define the set:

$$\begin{aligned} \Gamma(t) := \{\bar{t}_z \mid z \leq \bar{t}^+ \text{ with } \quad & W(\bar{s}, \bar{t}_z) := W(\bar{s}, \bar{t}) \text{ and} \\ & W(s, \bar{t}_z) := W(\bar{t}, \bar{s}) - z(\bar{s}) = (\bar{t}^+ - z)(\bar{s}) \text{ and} \\ & W(\bar{t}_z, s) := 0 \text{ and } W(\bar{t}_z, \bar{s}) := z(\bar{s}) \text{ and} \\ & W(p, \bar{t}_z) := W(\bar{t}_z, q) := 0 \text{ and} \\ & W(\bar{t}_z, p) := W(q, \bar{t}_z) := 1 \text{ for each } s \in S\} \end{aligned}$$

For later use let us write $\Gamma(T) := \bigcup_{t \in T} \Gamma(t)$ and likewise $\Gamma(\bar{T}) := \bigcup_{t \in T} \Gamma(\bar{t})$. Finally set $\vec{T} := \Gamma(T) \cup \Gamma(\bar{T})$.

Also define the following coding $\gamma : \Gamma(T) \cup \Gamma(\bar{T}) \to T$ by $\gamma(t_z) := t$ and $\gamma(\bar{t}_z) := t$ for each $t \in T$. Hence, $\gamma^{-1}(t) = \Gamma(t)$.

From the above definition one concludes (*):

$$\begin{aligned} &\forall m, m', m'' \in \mathbb{N}^{\vec{S}}\ \forall t \in T\ \forall t', t" \in \Gamma(t) : \\ &\quad (m[t'\rangle m' \text{ and } m[t''\rangle m'') \text{ implies} \\ &\quad (m'(i) - m'(i + |S| + 1) = m''(i) - m''(i + |S| + 1)) \text{ for each } 1 \leq i \leq 1 + |S| \end{aligned}$$

The same situation holds for transitions $\bar{t}'$ and $\bar{t}''$ in the set $\Gamma(\bar{t})$ which, too, are distinguished only by the index $z \leq \bar{t}^+$.

Transition t_z removes as many tokens from s as does t, but instead of placing $t^+(s)$ tokens on place s, it puts only $z(s)$ tokens and removes $t^+(s) - z(s)$ tokens from the corresponding place $\bar{s}$ in the reversed copy $\bar{N}$ of N. In this way a λ-transition with input places s and $\bar{s}$ removing $(t^+ - z)(s)$ tokens from each of these places afterwards is simulated without using any silent transition.

Note, $t_z^+(s) = z(s)$ for each $s \in S$ and by that an old transition $t \in T$ is extendable by the above construction to the transition $t_{t^+} \in \Gamma(t)$ which has the same effect — as regards its restriction on the subnet N — as had the former t.

Now we will formally prove that indeed $L(\vec{\Sigma}) = \text{twist}(L(\Sigma))$, if a marking is used as final marking which has only one token on place p which indicates that a transition sequence of even length must have been used.

Proof of $L(\vec{\Sigma}) \supseteq \text{twist}(L(\Sigma))$: Let $v = t_1 t_{2r} t_2 t_{2r-1} \cdots t_r t_{r+1}$ be the twist of an arbitrary transition sequence $t_1 t_2 t_3 \cdots t_{2r} \in L(\Sigma)$, i.e. , $v \in \text{twist}(L(\Sigma))$. We will inductively define a sequence $[w_i]_{i \in \mathbb{N}}$ of transition sequences $w_i \in \vec{T}^*$. These transition sequences will be shown to be firable but do not necessarily lead to a final marking.

Initially, w_0 will be defined as follows:

$$w_0 := \gamma^{-1}(v) \cap \{t_{z_1} \bar{t}'_{z_2} \mid\ t \in T, \bar{t}' \in \bar{T},\ z_1 := t^+, \text{ and } z_2 := \bar{t}'^+\}^*$$

Suppose, $w_i \in \vec{T}^*$ has already been defined, then w_{i+1} is obtained as follows: If m is the marking reachable by w_i from $\vec{m}_0$, i.e. , $\vec{m}_0[w_i\rangle m$ and $\exists j \leq |S|$ such that both $m(j) \geq 1$ and $m(j + |S| + 1) \geq 1$. In $w_i = t_{z_1} \cdots t_{z_k} \cdots t_{z_{2r}}$, here $t \in T \cup \bar{T}$, choose k maximal with $z_k(j) \geq 1$ and define $w_{i+1} := t_{z_1} \cdots t_{z_k - y} \cdots t_{z_{2r}}$, where y is an $|S|$-dimensional vector with $y(j) = 1$ and $y(r) = 0$ for $r \neq j$. If no such marking m exists then $w_{i+1} := w_i$.

The following properties hold for $[w_i]_{i \in \mathbb{N}}$:

1. The sequence $[w_i]_{i \in \mathbb{N}}$ becomes static after finitely many steps since $\vec{m}_0[w_i\rangle m_1$ and $\vec{m}_0[w_{i+1}\rangle m_2$ implies $m_1 \geq m_2$ and each entry of a marking is bounded below by zero.

2. If w_i is a transition sequence in $\vec{\Sigma}$ then also w_{i+1} is a transition sequence in $\vec{\Sigma}$ because the token generated in w_i and not in w_{i+1} is never used in the suffix following the modified transition t_{z_j}. In this suffix tokens are only removed from the place in question, so that the marking reached by occurrence of w_i tells us which tokens are not needed in this suffix.

3. $\vec{m}_0[w_i\rangle m_1$ and $\vec{m}_0[w_{i+1}\rangle m_2$ implies $m_1(i) - m_1(i+|S|+1) = m_2(1) - m_2(i+|S|+1)$ by the definition of $\vec{T}$, see (*) above.

Summarizing the facts about $[w_i]_{i\in\mathbb{N}}$ we observe: 1. shows, that after a finite number of steps a static transition sequence w' is reached such that $\vec{m}_0[w'\rangle m'$ and for all $i \leq 1+|S|$ one of $m'(i)$ or $m'(i+|S|+1)$ must equal zero, otherwise the inductive definition would still have changed w'. Since v was based on a valid transition sequence $t_1t_2t_3\cdots t_{2r}$ in Σ we know that $m_0[t_1\cdots t_r\rangle m''$ and $m''[t_{r+1}\cdots t_{2r}\rangle m_f$. It follows that w_0 is a transition sequence in $\vec{\Sigma}$, since both parts Σ and $\bar{\Sigma}$ of $\vec{\Sigma}$ work independently, only synchronized by toggling the marking of places p and q. We conclude that for $\vec{m}_0[w_0\rangle m'''$ the components i and $i+|S|+1$ of m''' must be equal for all $1 \leq i \leq |S|$. Due to 3., this then holds for all transition sequences in $[w_i]_{i\in\mathbb{N}}$. Except for p and q the corresponding components m''' therefore have to be zero. This means that for $\vec{m}_0[w'\rangle m'$ the marking m' reached must be equal to zero except for the place p, i.e. , $m' = \vec{m}_{\text{even}}$. Still $\gamma(w') = v$ holds, which is important for the extension of the proper coding h to the transitions in $\vec{T}$: if h is a coding used in Σ, even if it is the identity, then extend h to f: $\vec{T} \to h(T)$ by $f(t') := h(\gamma(t'))$ for each $t' \in \vec{T}$.

Proof of $L(\vec{\Sigma}) \subseteq \text{twist}(L(\Sigma))$: For an arbitrary transition sequence w in $\vec{\Sigma}$ $w = t_{z_1}\bar{t}_{z_{2r}}t_{z_2}\bar{t}_{z_{2r-1}}\cdots t_{z_r}\bar{t}_{z_{r+1}}$ such that $\vec{m}_0[w\rangle\vec{m}_{\text{even}}$ we have to verify that

$$\gamma(w) = t_1t_{2r}t_2t_{2r-1}\cdots t_rt_{r+1} = \text{twist}(v)$$

for some $v \in L(\Sigma)$. Obviously, $v = t_1t_2t_3\cdots t_{2r}$, and in order to prove that v is a legal transition sequence in $\vec{\Sigma}$ we modify the transitions in w as follows: Each transiton t_{z_j} is replaced by $t_j = t_{t_j^+}$ yielding the sequence w'. Still, $\gamma(w') = v$, by definition of γ and w' is a legal transition sequence in Σ that no longer reaches the final marking $\vec{m}_{\text{even}}$. This is obvious from the fact that $t^- = t_z^-$ for any transition $t \in T$ and any modification z, as long as the restriction of t_z^- on to the places of the subnet Σ within $\vec{\Sigma}$ are considered. The same holds true for the transitions $\bar{t}$ and $\bar{t}_z$ with respect to the subnet $\bar{\Sigma}$ within $\vec{\Sigma}$. By observation (*) it follows that the marking reached by w' with $\vec{m}_0[w'\rangle m$ is such, that (*) is still valid, which means that for each $1 \leq i \leq 1+|S|$ one has $m(i) - m(i+|S|+1) = 0$. But this means that the marking reached by $v_1 := t_1t_2\cdots t_r$ from m_0 in Σ is identical to that reached in $\bar{\Sigma}$ from $\bar{m}_0 = m_f$ by $v_2 := t_{2r}t_{2r-1}\cdots t_{r+1}$. Removing the toggling places p and q (together with their impinging arcs) from $\vec{\Sigma}$ would immediately show that v_1 is a transition sequence in Σ and v_2 one in $\bar{\Sigma}$. Thus, $v_1(v_2)^{\text{rev}} = v$ is a legal transition sequence in Σ with $m_0[v\rangle m_f$.

For those transition sequences that are of odd length, we can use the same construction and have to use the marking $\vec{m}_{\text{odd}}$. The formal verification is now left to the reader. $\square$

As an example let us construct the system $\vec{\Sigma}$ for the trivial net N having two places s and r and two transitions labeled a and b with initial marking $m_0(s) := 1$, $m_0(r) := 0$ and final marking $m_f(s) := 0$, $m_f(r) := 3$ as shown in Figure 2. The system $\vec{\Sigma}$ as it is defined by the construction in the proof of Theorem 5 is depicted in Figure 3. The reachability graph for the net shown in Figure 3 is displayed in Figure 4. What we still have to show, is how we can use this result to prove Theorem 3 and the generation of Petri net languages by reduction from linear context-free languages with the Dyck set D_1^*.

Figure 2: Net N before the construction.

Let $N := (S, T, W)$ be any Petri net. We will encode each marking $m \in \mathbb{N}^S$ reached from m_0 in a system $\Sigma := (N, m_0, m_f, h, V)$ by a string of opening brackets x of length $c(m)$. The encoding homomorphism $c : \mathbb{N}^S \to \mathbb{N} \setminus \{0\}$ maps the additive monoid $\mathbb{N}^S$ into the multiplicative monoid $\mathbb{N} \setminus \{0\}$ as follows:

$$\begin{aligned} c(\underline{0}) &:= 1 \\ c((m_1, m_2, \cdots, m_k + 1, \cdots, m_{|S|})) &:= c((m_1, m_2, \cdots, m_k, \cdots, m_{|S|})) \cdot p_k \end{aligned}$$

where p_k denotes the k^{th} prime number in the sequence $2, 3, 5, \ldots$. It is easily verified, see also [14, 15], that $c(m + m') = c(m) \cdot c(m')$. Now, we define a linear context-free grammar $G := (V_N, V_T, P, S)$ for any Petri net system Σ as follows:

Definition 0.5 *Let $\Sigma := (N, m_0, m_f, h, V)$ be any Petri net system based on the P/T-net $N := (S, T, W)$. The context-free grammar $G_\Sigma := (V_N, V_T, P, S)$ is given by the following productions of P, where capital letters are the nonterminals of V_N and $V_T := \{x, \bar{x}\} \cup V$ are the terminals.*

$$\begin{aligned} P \;:=\; & \{S \to x^{c(m_0)} A_t \mid \forall t \in T\} \\ \cup\; & \{S \to \lambda \mid \text{ if } m_0 = m_f\} \\ \cup\; & \{A_t \to \bar{x}^{c(t^-)} A_t \bar{x}^{c(t^+)}, A_t \to h(t) B_{t'} \mid \forall t, t' \in T\} \\ \cup\; & \{B_t \to x^{c(t^+)} B_t x^{c(t^-)}, B_t \to A_{t'} h(t) \mid \forall t, t' \in T\} \\ \cup\; & \{B_t \to h(t) x^{c(m_g)}, A_t \to \bar{x}^{c(m_g)} h(t) \mid \forall t \in T \; \forall m_g \in \mathbb{N}^S : m_g[t\rangle m_f\} \end{aligned}$$

$\square$

Using the construction in Definition 5 above the following can be shown:

Lemma 1 $L(G_\Sigma)\%D_1^* = \mathit{twist}^{-1}(L(\Sigma))$

Proof: First note that $\text{twist}^{-1}(abw) = a \cdot \text{twist}^{-1}(w) \cdot b$. It will thus be sufficient to look at a sequence of derivation steps in G_Σ that simulate two transitions in Σ. Suppose we

Figure 3: Net $\vec{N}$ after construction.

Figure 4: Reachability graph for $\vec{\Sigma}$.

are given a sentential form of G_Σ containing a substring $v = ax^iA_tb$ with $a, b \notin \{x, \bar{x}\}$, or $v = x^iA_ta$, and v is a prefix, or $v = ax^iA_t$, and v is a suffix.

Further, let $i = c(m)$ be the encoding of a marking as defined before. Then the only applicable productions are: $A_t \to \bar{x}^{c(t^-)}A_t\bar{x}^{c(t^+)}$, or $A_t \to h(t)B_{t'}$. Since $h \in \mathcal{H}$ is nonerasing, the first production has to be applied $n = \lfloor i/c(t^-)\rfloor$ times. Otherwise a derivation of a full Dyck string in D_1^* will not be possible. On the other hand i has to be a multiple of $c(t^-)$. Application of the productions then yield the string $\bar{x}^{c(t^+)\cdot n}$ to the right of the nonterminal A_t , where $n = c(m)/\bar{x}^{c(t^-)}$. Obviously $c(t^+) \cdot c(m)/\bar{x}^{c(t^+)} = c(m - t^- + t^+)$, which is the encoding of the marking reached in Σ from m by the occurrence of transition t. The reasoning for nonterminal $B_{t'}$ is similar but the terminal $h(t')$ will appear to the right of the symbol $B_{t'}$. This cycle can be iterated until a terminal production is applied. The sentential form generated by the initial rule of G_Σ encodes the initial marking of the net Σ. □

Now, this result enables us to complete the proof of Theorem 3: Let $L \in \underline{L}$ be any Petri net language specified by a Petri net system satisfying the normal form requirement of Definition 3, i.e. , using a net without selfloops or silent λ-transitions, having one initial and one final marking, $L := L(\Sigma_1)$, where $\Sigma_1 := (N, m_0, m_f, h, V)$ is such that $h : T^* \to V^*$ is a coding.

By Theorem 5 we find a new Petri net system Σ_2 such that

$$K := L(\Sigma_2) = \text{twist}(L(\Sigma_1)) = \text{twist}(L)$$

and applying Lemma 1 we find a linear context-free Grammar G such that

$$\begin{aligned} L(G)\%D_1^* &= \text{twist}^{-1}(K) \\ &= \text{twist}^{-1}(L(\Sigma_2)) \\ &= \text{twist}^{-1}(\text{twist}(L(\Sigma_1))) \\ &= L(\Sigma_1) \\ &= L \end{aligned}$$

□

Looking at the proof of Theorem 3 one realizes that the cancellation is solely done with the strings from the set EQ. Thus we could as well use either the cancellation $L(G)\%D_1'^*$ or $L(G)\%EQ$.

In [14] it was shown that $\underline{L} \subset \underline{Cf}\%D_1'^*$, and we have generalized this result by using the smaller family $\underline{Lin}$ of linear context-free languages. Theorem 2 showed that the class $\underline{Lin}\%EQ$ contains each recursively enumerable set. This characterization presumably is not valid for the class $\underline{Cf}\%D_1'^*$.

Proposition 1 *Previous work has shown:*

- *The emptiness problem is undecidable for the class* $\underline{Lin}\%D_1'^*$ *(see [15]).*
- *The word problem is decidable for the class* $\underline{Lin}\%D_1'^*$ *(see [15]).*

Still unknown, however, is the answer to the following question:

- Is the word problem decidable for the class $\underline{Cf}\%D_1'^*$?

In [15] a deterministic context-free grammar G is given that yields an NP-complete language as $L(G)\%D_1'^*$. This establishes a lower bound for the hardness of the word-problem.

Our second important result relies on the encoding of sentential forms and of counters using strings from the set $D_1'^*$ and will also work with D_1^*. The following lemma is only used for a simpler proof of Theorem 6 and shows, that under certain restrictions the leftmost derivations of a context-free grammar that lead to a terminal word, in which every subword from $\{x, \bar{x}\}^*$ belongs to a (possibly longer) word from $D_1'^*$, are uniquely determined by the so far generated substring from $\{x, \bar{x}\}^*$.

Lemma 2 *Let $G = (N, T, P, S)$ be a context-free grammar, $\{x, \bar{x}\} \subseteq T$, $V := N \cup T$. If for a nonterminal $A \in N$ there is a $k \in \mathbb{N}$ such that 1. through 3. below hold, then for any sentential form $wuAv$, $w \in (T \setminus \{x, \bar{x}\})^*$, $u \in \{x, \bar{x}\}^*$, $v \in V^*$, there is at most one sentential form $wuu'av'v$ derivable from $wuAv$ such that $uu' \in D_1'^*$:*

1. *$P \cap \{A \to v_1 A v_2 \mid v_1, v_2 \in V^*\}$ contains exactly one element, namely $A \to \bar{x}^k A w$, $w \in \{x\}^*$*
2. *for each n, $0 \leq n \leq k$, there is at most one production $A \to \bar{x}^n a w$, $a \in T \setminus \{x, \bar{x}\}$, $w \in V^*$*
3. *there are no productions other than those mentioned in 1. or 2. with a lefthand side A.*

Proof: Suppose to the contrary, that there are two distinct sentential forms that could be derived from $wuAv$, say wuu_1av_1v and wuu_2bv_2v, such that uu_1 and uu_2 are in $D_1'^*$. Then $u_1 = u_2$ since all productions of G with left-hand side A derive $\bar{x}$ to the left of A. All terminating productions derive less than k symbols $\bar{x}$, so $A \to \bar{x}^k Aw$ has to be applied $|u_1|$div k times, and then only one production may be used. Thus the leftmost derivation of $wuAv$ is uniquely determined. $\square$

Definition 0.6 *Let $\mathcal{L}$, $\mathcal{K}$ be families of languages, then the* wedge *operation $\wedge$ is defined by*

$$\mathcal{L} \wedge \mathcal{K} := \{L \mid L = M \cap K \text{ for some } M \in \mathcal{L} \text{ and } K \in \mathcal{K}\}$$

$\square$

The following theorem shows the relation between the class $\underline{Cf}^{\triangle}\%D_1^*$ and the closure of intersections of λ-free context-free languages with respect to nested iterated substitution.

Theorem 6 $\tau^{\infty}(\mathcal{H}(\underline{Cf}^{\triangle} \wedge \underline{Cf}^{\triangle})) \subseteq \underline{Cf}\%D_1^*$

Proof: As a simplification we omit the non-erasing homomorphism, but it will become clear that we could by introducing more productions include it into the construction. We will now prove by induction a slightly stronger version of the above theorem: for every language $L \in \tau^\infty(\underline{Cf}^\triangle \wedge \underline{Cf}^\triangle)$ there is a context-free grammar G with $L(G)\%D_1^* = L$ and G derives no terminal string that ends or begins with a parenthesis symbol. To this end we will use induction on the generation of languages by application of nested iterated substitution starting with the intersections of context-free languages. For the basis of the induction let $L_1 = L(G_1)$ and $L_2 = L(G_2)$ be context-free languages, G_1 and G_2 are context-free grammars. Without loss of generality be $G_1 = (N_1, T_1, P_1, S_1)$ in Chomsky normal form and $G_2 = (N_2, T_2, P_2, S_2)$ in the following version of the super normal form from [19]:

$$P \subseteq \{A \to aBC \mid A, B, C \in N_2, a \in T_2\} \cup \{A \to w \mid A \in N_2, w \in T_2^+\}$$

where we may suppose further that $(T_1 \cup T_2) \cap \{x, \bar{x}\} = \emptyset$ and $\lambda \notin L_1 \cup L_2$.

In the sequel we will step by step construct a context-free grammar $G := (N, T, P, S)$, that generates $L = L_1 \cap L_2 = L(G)\%D_1^*$. For every string $w \in L$ with less than 3 characters P will contain a production $S \to w$. For every sentential form $ABC \in N^*$ that is derivable from S_1 and contains exactly 3 nonterminal symbols there is a rule $S \to L_A B R_C$ in P. The symbols L_A and R_C serve as end markers that prevent G from generating x's and $\bar{x}$'s at the end or beginning of a sentential form. The terminating productions of L and R symbols will generate encodings of initial and final pushdown contents resp., according to the encoding described below. The remaining productions are divided into two groups: the first simulate productions of G_1 and do not generate terminal symbols directly but derive an H, J or K symbol whenever G_1 would yield a terminal symbol. The nondeterministic choice of H, J or K depends on how G_2 yields a terminal symbol, let us call it a, at this point. $K_{B \to aw}$ means a production $B \to aw$ is applied, $J_{B \to aCD}$ means a production $B \to aCD$ is applied and H_a means that the symbol a is generated by an application of a production $A \to vaw$ to the left of a.

The remaining productions of G make sure, that the choices of the above symbols were made correctly by keeping a pushdown stack of yet unprocessed symbols encoded by parentheses. We set $k := |N_2| + |T_2| + 1$ and define a mapping $f : (T_2 \cup N_2)^* \to \mathbb{N}$ by:

$$\begin{aligned} f(\lambda) &:= 0 \\ f(t_i w) &:= |N_2| + i + k \cdot f(w) \text{ for } t_i \in T_2,\ 1 \leq i \leq |T_2| \\ f(A_j w) &:= j + k \cdot f(w) \text{ for } A_j \in N_2,\ 1 \leq j \leq |N_2| \end{aligned}$$

We are now prepared for the set of productions of P:

$$
\begin{array}{rcl}
P & := & \{S \to w \mid |w| \leq 2\} \\
 & \cup & \{S \to L_A B R_C \mid ABC \text{ is derivable from } S_1\} \\
 & \cup & \{A \to BC \mid A \to BC \in P_1\} \\
 & \cup & \{A \to H_a \mid A \to a \in P_1\} \\
 & \cup & \{A \to J_{B \to aCD} \mid A \to a \in P_1, B \to aCD \in P_2\} \\
 & \cup & \{A \to K_{B \to aw} \mid A \to a \in P_1, B \to aw \in P_2\} \\
 & \cup & \{L_A \to L_B C \mid A \to BC \in P_1\} \\
 & \cup & \{R_A \to B R_C \mid A \to BC \in P_1\} \\
 & \cup & \{L_A \to a x^{f(BC)} \mid A \to a \in P_1, S_2 \to aBC \in P_2\} \\
 & \cup & \{R_A \to \bar{x}^{f(E)} a \mid A \to a \in P_1, E \to a \in P_2\} \\
 & \cup & \{R_A \to \bar{x}^{f(a)} a \mid A \to a \in P_1\} \\
 & \cup & \{J_{B \to aCD} \to \bar{x} J_{B \to aCD} x^{k^2} \mid B \to aCD \in P_2\} \\
 & \cup & \{J_{B \to aCD} \to \bar{x}^{f(B)} a x^{f(CD)} \mid B \to aCD \in P_2\} \\
 & \cup & \{H_a \to \bar{x}^k H_a x \mid a \in T_1\} \\
 & \cup & \{H_a \to \bar{x}^{f(a)} a \mid a \in T_1\} \\
 & \cup & \{K_{B \to aw} \to \bar{x}^k K_{B \to aw} x^{k|w|} \mid B \to aw \in P_2\} \\
 & \cup & \{K_{B \to aw} \to \bar{x}^{f(B)} a x^{f(w)} \mid B \to aw \in P_2\}
\end{array}
$$

Inspection of P shows, that the first part indeed simulates G_1 and the productions with a left-hand side J, H or K conform with the premise of Lemma 2. Taking into account, that each L symbol generates encodings of initial sentential forms only, and R symbols yield encodings of final sentential forms, it is thus inductively guaranteed by Lemma 2, that the leftmost derivation of a string consisting entirely of J, H and K symbols is uniquely determined if it is to derive complete Dyck subwords. It will yield a terminal string if and only if the choice of productions is correct.

Obviously no parentheses are ever generated at the edges of sentential forms. This concludes the basis of the proof. For the inductive step suppose, that a language L on an n element alphabet and n languages $L_1, L_2, \ldots, L_n$ are given, $a_i \in L_i$ for all terminal symbols a_i of L, $1 \leq i \leq n$. Further $\{L, L_1, \cdots, L_n\} \subset \tau^\infty(\underline{Cf}\%D_1^*)$ and there are grammars of the above type for all $n+1$ languages by the inductive hypothesis, $L = L(G)\%D_1^*$, $G = (N, T, P, S)$, $L = L(G_i)\%D_1^*$.

For the nested iterated substitution τ given by : $\tau(a_i) := L_i$ for every terminal symbol a_i of L we will describe a grammar for $\tau^\infty(L)$. Without loss of generality we suppose, that the nonterminal symbols of the grammars generating the above languages L_i are disjoint. Now, omitting further details, the grammar $G_i := (N_t, T_t, P_t, S)$ for $\tau^\infty(L)$ is given by: $N_t := N \cup N_1 \cup N_2 \cdots N_n$ $T_t := T_1 \cup \cdots \cup T_n$, and $P_t := P' \cup P_1 \cup P_2 \cdots \cup P_n$, where P' consists of the productions P, modified in such a way, that on the right side of each production terminal symbols are replaced by starting symbols of the appropriate grammar for $L_1, \ldots, L_n$. Finally we observe that terminal symbols may be replaced by nonempty strings without disturbing the simulations. This proof equally works for the one-sided reduction with $D_1'^*$. $\square$

Corollary 3

$$\mathcal{H}(\underline{Cf} \wedge \underline{Cf}) \subseteq \underline{Cf}\%D_1'^*$$

$$\mathcal{H}(\underline{Cf} \wedge \underline{Cf}) \subseteq \underline{Cf}\%D_1^*$$

Proof: By Theorem 6 we have $\underline{Cf}^\triangle \wedge \underline{Cf}^\triangle \subseteq \underline{Cf}\%D_1^*$ for the trivial substitution of a terminal symbol with itself. Should the intersection of two languages contain λ, then we include the production $S \to \lambda$ into the context-free grammar in which the Dyck strings are to be cancelled. $\square$

Note, that the inclusion $\underline{Cf}^\triangle \wedge \underline{Cf}^\triangle \subset \underline{Cf}\%D_1^*$ is proper. To see this consider the language

$$E := \{a^m \mid m = 2^n, n \in I\!N\}$$

which is obtained as $L(G_E)\%D_1^* : G_E := (N_E, T_E, P_E, S_E)$, where $N_E := \{S_E, L, R, A\}$, $T_E := \{a, x, \bar{x}\}$, and $P_E := \{S \to LAR, L \to Lx, L \to x, R \to \bar{x}R, R \to \bar{x}, A \to \bar{x}AAx, A \to a\}$.

If E were the intersection of two context-free languages L_1 and L_2 then $E = L_1 \cap L_2 = (L_1 \cap \{a\}^*) \cap (L_2 \cap \{a\}^*) = R$ for a regular set R, since $\underline{Cf}$ is closed with respect to intersection with regular sets and a context-free language on a singular alphabet is known to be itself regular. E is obviously not regular, so $E \notin \underline{Cf} \wedge \underline{Cf}$.

The family $\underline{Cf}\%D_1^*$ has many interesting properties that follow from [14, Theorem 4] and from Theorem 6.

Corollary 4 *The intersection of any two languages from the family* $\underline{Cf} \cup \underline{L}$ *is contained in each of the following families:* $\underline{Cf}\%D_1^*$, $\underline{Cf}\%D_1'^*$, *or* $\underline{Cf}\%EQ$.

From Theorem 6 we may as well conclude Corollary 5 in [15], immediately:

Corollary 5 *The emptiness problem of neither* $\underline{Cf}\%D_1^*$, $\underline{Cf}\%D_1'^*$, *nor of* $\underline{Cf}\%EQ$ *is decidable.*

Conclusion

Not only $\tau^\infty(\mathcal{H}(\underline{L} \wedge \underline{Cf}))$, by the results in [15], but also the class $\tau^\infty(\mathcal{H}(\underline{Cf}^\triangle \wedge \underline{Cf}^\triangle))$ is obtainable by one-sided or two-sided Dyck reductions from the family of context-free languages.

We have generalized the result from [14] by showing $\underline{L} \subseteq \underline{Lin}\%D_1^*$, see also [16]. We were able to show, that the word problem for the class $\underline{Cf}\%D_1^*$ is decidable, is undecidable for $\underline{Cf}\%EQ$, while this is still an open question for $\underline{Cf}\%D_1'^*$.

Acknowledgements

We are indebted to the anonymous referees of the 12^{th} Intern. Colloquium on Petri Nets for their proposals for improvements. We would like to thank Manfred Kudlek and also all other members of the working group TGI for their vivid discussions on the presented subject.

References

[1] M. Benois: *Parties rationelles du groupe libre.* C. R. Acad. Sc. Paris, Ser. A t. 269 (1969) 1188–1190.

[2] R.V. Book, M. Jantzen, C. Wrathall: *Monadic Thue systems.* Theoret. Comput. Sci. 19 (1982) 231–251.

[3] F.J. Brandenburg: *Cancellations in linear context-free languages.* Techn. report MIP-8904, Univ. Passau (1989).

[4] F.J. Brandenburg, J. Dassow: *Reductions of picture words.* Techn. report MIP-8905, Univ. Passau (1989).

[5] J. Dassow, G. Paun: *Regulated rewriting in formal language theory.* EATCS Monographs 18, Springer-Verlag (1989).

[6] Ch. Frougny, J. Sakarovitch, P. Schupp: *Finiteness conditions on subgroups and formal language theory.* Proc. London Math. Soc. 58 (1989) 74–88.

[7] V. Geffert: *Grammars with context dependency restricted to synchronization.* Lecture Notes in Comput. Sci. vol. 233, Springer-Verlag (1986) 370–378.

[8] V. Geffert: *Normal forms for phrase-structure grammars.* RAIRO Theoretical Informatics and Applications, 25 (1991) 473–496.

[9] S.A. Greibach: *Full AFLs and nested iterated substitution.* Inform. Control 16 (1970) 7–35.

[10] T.V. Griffiths: *Some remarks on derivations in general rewriting systems.* Inform. Control 12 (1968) 27–45.

[11] M. Hack: *Petri net languages.* C.S.G. Memo 124, Project MAC, MIT (1975).

[12] M. Jantzen: *Language theory of Petri nets.* Lecture Notes in Comput. Sci., vol 254 (1987) 397–412.

[13] M. Jantzen: *Confluent string rewriting.* EATCS Monographs 14, Springer-Verlag (1988).

[14] M. Jantzen, H. Petersen: *Petri net languages and one-sided $Dyck_1$-reductions of context-free sets.* in K. Voss, H. Genrich, G. Rozenberg (eds.): Concurrency and nets. Springer-Verlag (1987) 245–252.

[15] M. Jantzen, M. Kudlek, K.-J. Lange, H. Petersen: *$Dyck_1$-reductions of context-free languages.* Computers and Artificial Intelligence, 9 (1990) 3–18.

[16] M. Jantzen, H. Petersen: *Twisting Petri net languages and how to obtain them by reducing linear context-free sets.* In: Proc. 12^{th} Intern. Conf. on Petri nets, Gjern (1991) 228–236.

[17] L. Kari: *On Insertion and Deletion in Formal Languages.* Dissertation, University of Turku (1991).

[18] T. Kimura: *Formal description of communication behaviour.* Proc. Johns Hopkins Conf. on Information Sciences and Systems (1979).

[19] H.A. Maurer, A. Salomaa, D. Wood: *A super normal form theorem for context-free grammars.* McMaster Univ., Comp. Science Techn. Report 80-CS-20 (1980).

[20] H.A. Maurer, G. Rozenberg, E. Welzl: *Using string languages to describe picture languages.* Inform. Control 54 (1982), 155–185.

[21] J.L. Peterson: *Petri net theory and the modeling of systems.* Prentice Hall (1981).

[22] W.J. Savitch: *How to make arbitrary grammars look like context-free grammars.* SIAM J. Comput. 2 (1973) 174–182.

[23] W.J. Savitch: *Some characterizations of Lindenmayer systems in terms of Chomsky-type grammars and stack machines.* Inform. Control 27 (1975) 37–60.

[24] W.J. Savitch: *Parenthesis grammars and Lindenmayer sytems.* In: G. Rozenberg, A. Salomaa (eds.): The Book of L, Springer-Verlag (1986) 403–411.

[25] D. Stanat: *Formal languages and power series.* 3^{rd} ACM Sympos. Theory of Computing (1971) 1–11.