Fachbereich Informatik der Universität Hamburg
Vogt-Kölln-Str. 30, D-22527 Hamburg / Germany
University of Hamburg — Computer Science Department

Bericht Nr. 204 • Report No. 204

On Twist-Closed Trios

Matthias Jantzen

FBI-HH-B-204/97
August 1997

Vorgeschlagen durch Recommended by

Prof. Dr. Rüdiger Valk and Prof. Dr. Christian Freksa

Abstract

The language theoretic operation *twist* from [JaPe 87] is studied in connection with the semiAFLs of languages accepted by reversal bounded multipushdown and multicounter acceptors. It is proved that the least *twist*-closed trio generated by $MIR := \{ww^{rev} \mid w \in \{a,b\}^*\}$ is equal to the family of languages accepted in quasi-realtime by nondeterministic one-way multipushdown acceptors which operate in such a way that in every computation each pushdown makes at most one reversal. Thus, $\mathcal{M}_\cap(MIR) = \mathcal{M}_{twist}(MIR)$ and this family is principal both as a *twist*-closed and as an *intersection*-closed semiAFL. This is in contrast to the semiAFL of languages accepted by reversal-bounded multicounter machines in quasi-realtime. This family is a well known semiAFL which is principal as an *intersection*-closed semiAFL with generator $B_1 := \{a_1^n \overline{a}_1^n \mid n \in I\!N\}$, see [Grei 78], but is not principal as a semiAFL. It is here shown that it forms a hierarchy of *twist*-closed semiAFLs and therefore cannot be principal as *twist*-closed semiAFL.

Zusammenfassung

Die in [JaPe 87] definierte Operation *twist* auf Wörtern und formalen Sprachen wird untersucht in Verbindung mit den semiAFL's der von umkehrbeschränkten Keller- und Zählerautomaten akzeptierten Sprachen. Es wird gezeigt, daß das kleinste, von der Sprache $MIR := \{ww^{rev} \mid w \in \{a,b\}^*\}$ generierte *twist*-abgeschlossene Trio identisch ist mit der Familie aller Sprachen, die von mehr-Kellerautomaten in quasi-Realzeit mit umkehrbeschränkten Kellern akzeptiert werden. Daher ist $\mathcal{M}_\cap(MIR) = \mathcal{M}_{twist}(MIR)$ als *durchschnitts-* und als *twist*-abgeschlossenes Trio eine sogenannte Haupt-semiAFL (*principal semiAFL*). Im Unterschied dazu ist die Familie $\mathcal{M}_\cap(B_1)$, mit Generator $B_1 := \{a_1^n \overline{a}_1^n \mid n \in I\!N\}$, der in quasi-Realzeit von umkehrbeschränkten mehr-Zählerautomaten akzeptierten Sprachen, vergl. [Grei 78], zwar *twist*-abgeschlossen, aber es gilt nicht $\mathcal{M}_\cap(B_1) = \mathcal{M}_{twist}(B_1)$.

1 Introduction

In connection with a representation of Petri net languages by Dyck-reductions of (linear) context-free sets the operation *twist* was defined and used for the first time, see [JaPe 87, JaPe 94]. The definition of this new language theoretic operation is based upon a mapping from strings to strings which rearranges letters: for a string $w := x_1 x_2 \cdots x_{n-1} x_n$ the unique new string is $twist(w) := x_1 x_n x_2 x_{n-1} \cdots x_{\lfloor \frac{n}{2} \rfloor + 1}$. Observe, that $twist : \Sigma^* \longrightarrow \Sigma^*$ is a bijection and it's inverse mapping $twist^{-1}(w)$ yields a unique string v with $twist(v) = w$. It follows that for each string w there exists a non-negative integer $k \in I\!N$ such that $twist^k(w) = w$. The mapping *twist* can also be regarded as a permutation of the n distinct positions for the symbols of a string of length n. As language theoretic operation *twist* is generalized to languages and families of languages in the obvious way, see Def.4.

It was shown in [JaPe 94], Th.2.10, that the family $\mathcal{R}eg$ of regular sets is closed with respect to *twist*. The inclusion $twist(\mathcal{R}eg\ ) \subsetneq \mathcal{R}eg$ must be proper since $twist(MIR) = \{a^2, b^2\}^*$, where $MIR := \{ww^{rev} \mid w \in \{a,b\}^*\}$ is the non-regular context-free set of palindroms of even length. This observation means, that the regular set $\{a^2, b^2\}^*$ will never appear as $twist(R)$ for any regular set $R \in \mathcal{R}eg$. Notice, $twist^{-1}(MIR) = COPY := \{ww \mid w \in \{a,b\}^*\}$. In [JaPe 94], Th.2.11, it was also proved that the family $\mathcal{L}_0$ of λ-free labelled, terminal Petri net languages is closed with respect to the operation *twist*. Again, $twist(\mathcal{L}_0) \subsetneq \mathcal{L}_0$, since $twist(MIR) \in \mathcal{L}_0$ but $MIR \notin \mathcal{L}_0$ follows from results in [Grei 78, Jant 79a] using the decidability of the reachability problem for Petri nets, proved in [Mayr 84, Kosa 82]. The proof for the closure of $\mathcal{L}_0$ under *twist* essentially uses a construction that can easily be be modified for showing closure with respect to the operation $\frac{1}{2}$. In fact, the two operations $\frac{1}{2}$ and *reversal* are closely linked with the operation *twist*. In conjunction with the usual trio-operations the combination of *twist* and *product* can simulate any (finite) combination of *intersection*, *reversal*, and $\frac{1}{2}$ as shown below in Lemma 1 and in Theorem 2.11 where a new morphic characterization of the recursively enumerable sets is given by $\mathcal{R}e = \hat{\mathcal{H}}(\mathcal{H}^{-1}(twist(lin\,\mathcal{C}f))) = \hat{\mathcal{M}}_{twist}(dMIR)$. Similar results are known for principal *intersection*-closed full trios (see [BaBo 74]) and for full principal trios, the generator of which is as rich in structure as the twinshuffle language L_{TS} (see [Salo 81], Chapt. 6, for a condensed presentation. L_{TS} was there abbreviated by L_Σ). A similar result has been shown by Engelfriet in [Enge 96] for the reverse twin shuffle language L_{RTS}. However, neither L_{TS} nor L_{RTS} is context-free.

We will prove, Theorem 2.11, the characterization of the r.e. sets directly in a manner similar to proofs in [BaBo 74] which nowadays can be viewed as standard methods.

It was proved in[BoNP 74] that the non-full *intersection*-closed trio $\mathcal{M}_\cap(MIR)$ equals $\mathcal{H}(lin\,\mathcal{C}f \wedge lin\,\mathcal{C}f \wedge lin\,\mathcal{C}f)$, where for families of languages $\mathcal{K}$ and $\mathcal{L}$ the wedge is defined by $\mathcal{K} \wedge \mathcal{L} := \{K \cap L \mid K \in \mathcal{K} \text{ and } L \in \mathcal{L}\}$. We will show here that the closure of $lin\,\mathcal{C}f$ under *twist* contains generators for the family $\mathcal{M}_\cap(MIR)$ and that the latter family of quasi-realtime multipushdown languages is closed with respect to the operation *twist*, hence $\mathcal{M}_\cap(MIR) = \mathcal{M}_{twist}(MIR)$.

The situation becomes different for the semiAFL of languages accepted by one-way reversal-bounded multicounter machines in quasi-realtime. This family is a well known semiAFL which is principal as an *intersection*-closed semiAFL with generator $B_1 := \{a_1^n \overline{a}_1^n \mid n \in I\!N\}$, which is not a principal semiAFL, see [FiMR 68, Grei 78]. $\bigcup\limits_{i \geq 0} \mathcal{M}(C_i) =$

$\mathcal{M}_\cap(C_1) = BLIND(lin) = \bigcup_{i\geq 0} \mathcal{M}(B_i) = \mathcal{M}_\cap(B_1) = RBC(n)$ is know from [Grei 78]. The known situation for these hierarchies is as follows: For all $i \geq 1$ we have $\mathcal{M}(B_i) \subsetneq \mathcal{M}(B_{i+1})$, see [Gins 75], $\mathcal{M}(C_i) \subsetneq \mathcal{M}(C_{i+1})$, and $\mathcal{M}(B_i) \subseteq \mathcal{M}(C_i)$, shown in [Grei 76, Grei 78], [Latt 77], [Latt 78, Latt 79] and [Kort 80]. We will in addition show here that $\bigcup_{i\geq 0} \mathcal{M}(C_i) = \mathcal{M}_\cap(C_1)$ forms a hierarchy of *twist*-closed semiAFLs and therefore cannot be principal as *twist*-closed semiAFL. Each such semiAFL $\mathcal{M}(C_k)$ can be characterized as family $BLIND(n) = BLIND(lin)$ of languages acceptable by blind k-counter machines in quasi-realtime (or equivalently linear time), [Grei 78]. Slightly improving a construction by Greibach we will prove that the semiAFL $\mathcal{M}(C_k)$ of blind k-counter languages is (strictly) included in the family $\mathcal{M}(B_{k+1})$ of languages accepted in quasi-realtime by nondeterministic one-way (k+1)-counter acceptors which operate in such a way that in every computation each counter makes at most one reversal.

2 Basic Definitions

2.1 Definition

Let $\mathcal{R}eg$ (resp. $lin\mathcal{C}f$, $\mathcal{C}f$, $\mathcal{L}_0$, $\mathcal{C}s$, $\mathcal{L}_0^\lambda$, $\mathcal{R}ec$, $\mathcal{R}e$) denote the families of regular sets (linear context-free, context-free, λ-free labelled terminal Petri net, context sensitive, arbitrarily labelled terminal Petri net, recursive, and recursively enumerable languages, respectively).

$\square$

2.2 Definition

Let $w_1, w_2 \in \Sigma^*, w_1 := x_1 x_2 \cdots x_m$, and $w_2 := y_1 y_2 \cdots y_n$ where $x_i, y_j \in \Sigma$ for $1 \leq i \leq m$ and $1 \leq j \leq n$.

Then the *shuffle* $\sqcup\!\sqcup$ and the *literal shuffle* $\sqcup\!\sqcup_{lit}$ are defined as follows:

$$w_1 \sqcup\!\sqcup w_2 := \left\{ u_1 v_1 u_2 v_2 \cdots u_n v_n \;\middle|\; \begin{array}{l} n \in I\!N, u_i, v_i \in \Sigma^*, w_1 = u_1 u_2 \cdots u_n, \\ w_2 = v_1 v_2 \cdots v_n \end{array} \right\},$$

$$w_1 \sqcup\!\sqcup_{lit} w_2 := \begin{cases} x_1 y_1 x_2 y_2 \cdots x_m y_m y_{m+1} \cdots y_n\,, & \text{if } m \leq n \\ x_1 y_1 x_2 y_2 \cdots x_n y_n x_{n+1} \cdots x_m\,, & \text{if } n < m \end{cases}$$

$\square$

2.3 Definition

Specific languages we consider are constructed using the alphabets $\Gamma := \{a, b\}$, $\overline{\Gamma} := \{\overline{a}, \overline{b}\}$ and $\Gamma_n, \overline{\Gamma}_n$ specified for each $n \in I\!N, n \geq 1$ by: $\Gamma_n := \{a_i, b_i \mid 1 \leq i \leq n\}$, $\overline{\Gamma}_n := \{\overline{a}_i, \overline{b}_i \mid 1 \leq i \leq n\}$, and the homomorphisms $^-, h, \overline{h}$, and h_i defined by:

$\overline{x} := \begin{cases} \overline{x} & , \text{if } x \in \Gamma \\ x & , \text{if } x \in \overline{\Gamma} \end{cases}$, $h(x) := \begin{cases} x & , \text{if } x \in \Gamma \\ \lambda & , \text{if } x \in \overline{\Gamma} \end{cases}$, $\overline{h}(x) := \begin{cases} \lambda & , \text{if } x \in \Gamma \\ x & , \text{if } x \in \overline{\Gamma} \end{cases}$, and $h_i(a_1) := a_i, h_i(b_1) := b_i$ for $i \in I\!N \setminus \{0\}$,

By $|w|_x$ we denote the number of occurences of the symbol $x \in \Sigma$ within the string $w \in \Sigma^*$ and $|w| := \Sigma_{x \in \Sigma} |w|_x$ is the length of w.

$$\begin{array}{rcl}
B_i & := & \begin{cases} B_{i-1} \sqcup \{a_i^n b_i^n \mid n \in I\!N\} & , \text{if } i \geq 2 \\ \{a_1^n b_1^n \mid n \in I\!N\} & , \text{if } i = 1 \end{cases} \\
C_i & := & \begin{cases} C_{i-1} \sqcup h_i(C_1) & , \text{if } i \geq 2 \\ \{w \in \{a_1, b_1\}^* \mid |w|_{a_1} = |w|_{b_1}\} & , \text{if } i = 1 \end{cases} \\
D_i & := & \begin{cases} D_{i-1} \sqcup h_i(D_1) & , \text{if } i \geq 2 \\ \{w \in \{a_1, b_1\}^* \mid |w|_{a_1} = |w|_{b_1}, & \\ \quad \text{and } \forall w = uv : |u|_{a_1} \geq |u|_{b_1}\} & , \text{if } i = 1 \end{cases} \\
E_i & := & \{a_1^n a_2^n \cdots a_i^n \mid n \in I\!N\} \\
dMIR & := & \{wcw^{rev} \mid w \in \Gamma^*\}. \\
MIR & := & \{ww^{rev} \mid w \in \Gamma^*\}. \\
PAL & := & \{w \mid w = w^{rev}, w \in \Gamma^*\}. \\
dCOPY & := & \{wcw \mid w \in \{a, b\}^*\} \\
COPY & := & \{ww \mid w \in \Gamma^*\} \\
L_{TS} & := & \{w \in (\Gamma \cup \overline{\Gamma})^* \mid \overline{h(w)} = \overline{h}(w)\} \\
L_{RTS} & := & \{w \in (\Gamma \cup \overline{\Gamma})^* \mid \overline{h(w)} = \overline{h}(w^{rev})\} \\
twinPAL & := & \{w \in (\Gamma \cup \overline{\Gamma})^* \mid \overline{h(w)} \in MIR \text{ and } \overline{h}(w) \in \overline{MIR}\}
\end{array}$$

□

Let us repeat the basic notions and results from AFL-theory details of which are to be found in the textbooks of Ginsburg, [Gins 75], and Berstel, [Bers 80].

A family of languages $\mathcal{L}$ is called trio if it is closed under inverse homomorphisms, intersection with regular sets, and nonerasing homomorphisms. The least trio containing the family $\mathcal{L}$ is written $\mathcal{M}(\mathcal{L})$. If $\mathcal{L} := \{L\}$, then L is a generator of the trio $\mathcal{M}(\mathcal{L})$, shortly written as $\mathcal{M}(L)$ and then called principal. A union-closed trio is called semiAFL. Any principal trio is closed with respect to union and thus forms a semiAFL. If a trio is closed under arbitray homomorphisms, then it is called a full trio, written $\hat{\mathcal{M}}(\mathcal{L})$.

A family of languages $\mathcal{L}$ is called an AFL (or full AFL) if it is a trio (full trio, resp.) which is closed under the operations union, product and Kleene plus. The smallest AFL (or full AFL) containing the family $\mathcal{L}$ is written $\mathcal{F}(\mathcal{L})$ ($\hat{\mathcal{F}}(\mathcal{L})$, resp.). Each full AFL is closed with respect to Kleene star.

If a trio $\mathcal{M}(\mathcal{L})$ (or an AFL $\mathcal{F}(\mathcal{L})$) is in addition closed with respect to one further operation $\circledast$ then this family will be called $\circledast$-closed and abbreviated as $\mathcal{M}_{\circledast}(\mathcal{L})$ (resp. $\mathcal{F}_{\circledast}(\mathcal{L})$).

The language D_1 defined above is the so-called semi-Dyck language on one pair of brackets which is often abbreviated by D'^*_1, see e.g. [Latt 77, Latt 79] or [Bers 80]. D_n here denotes the n-fold shuffle of disjoint copies of the semi-Dyck language D_1 and it is known [Grei 78] that

$$\bigcup_{i\geq 0} \mathcal{M}(D_i) = \mathcal{M}_\cap(D_1) = PBLIND(n).$$

The latter family is the family of languages accepted in quasi-realtime by nondeterministic one-way multicounter acceptors which operate in such a way that in every computation no counter can store a negative value, and whether or not the value stored in a counter is *zero* cannot be used for deciding the next move. This family is equal to the family $\mathcal{L}_0$ of λ-free labelled terminal Petri net languages, see [Jant 79a, Grei 78].

The languages C_n are the Dyck languages on n pairs of brackets $a_i, \overline{a}_i$, often abbreviated by D^*_n, see again [Latt 77, Latt 79] or [Bers 80]. Greibach, [Grei 78], has shown that

$$\bigcup_{i\geq 0} \mathcal{M}(C_i) = \mathcal{M}_\cap(C_1) = BLIND(lin) = \bigcup_{i\geq 0} \mathcal{M}(B_i) = \mathcal{M}_\cap(B_1) = RBC(n).$$

Here $BLIND(lin)$ denotes the family of languages accepted in linear time by nondeterministic one-way multicounter acceptors which operate in such a way that in every computation all counters may store arbitrary integers, and the information on the contents of the counters cannot be used for deciding the next move. The family $RBC(n)$ is the family of languages accepted by nondeterministic one-way multicounter acceptors performing at most one reversal in each accepting computation. Hromkovič proved in [Hrom 85] that quasi-realtime multicounter machines and quasi-realtime partially blind multicounter machines with a constant number of reversals define the same family of languages, see also [Jant 79b] and [ĎuHr 87].

The least *intersection*-closed full semiAFL $\hat{\mathcal{M}}_\cap(B_1)$ has been characterized in [BaBo 74] as the family of languages accepted by nondeterministic on-line multicounter acceptors which operate in such a way that in every computation each counter makes at most one reversal. It was there shown that this class contains only recursive sets, i.e. $\hat{\mathcal{M}}_\cap(B_1) \subseteq \mathcal{R}ec$. Latteux has shown in [Latt 77] that $\mathcal{M}_\cap(B_1)$ is the smallest *commutation*-closed trio, i.e., if $L \in \mathcal{M}_\cap(B_1)$ then $\psi(\psi^{-1}(L)) \in \mathcal{M}_\cap(B_1)$, where ψ is the Parikh mapping $\psi : \Sigma^* \longrightarrow I\!N^{|\Sigma|}$ defined by $\psi(w) := (|w|_{x_1}, \ldots, |w|_{x_{|\Sigma|}})$. Latteux, [Latt 79], has also observed that $\hat{\mathcal{M}}_\cap(B_1) = \mathcal{M}_\cap(B_1) \subsetneq \mathcal{H}(lin\mathcal{C}f \wedge lin\mathcal{C}f \wedge lin\mathcal{C}f)$.

dMIR, *MIR* and *PAL* are well-known context-free generators of the family $lin\mathcal{C}f$ of linear context-free languages: $lin\mathcal{C}f = \mathcal{M}(dMIR) = \mathcal{M}(MIR) = \hat{\mathcal{M}}(PAL)$. These languages are precisely the languages accepted by nondeterministic on-line single pushdown acceptors which operate in such a way that in every accepting computation the pusdown store makes at most one reversal. And this family is not closed with respect to product or Kleene plus.

Similarly, the *intersection*-closed semiAFL $\mathcal{M}_\cap(MIR)$ can be identified with the family of languages accepted by nondeterministic on-line multipushdown acceptors which operate in such a way that in every computation each pushdown makes at most one reversal and that work in quasi-realtime, see [BoGr 70]. This family, however, becomes the set of recursively enumerable languages if erasing is allowed and was characterized in [BaBo 74] by $\mathcal{R}e = \hat{\mathcal{M}}_{\cap}(MIR) = \hat{\mathcal{M}}(twinPAL)$.

2.4 Definition

Let Σ be an alphabet, then $twist : \Sigma^* \longrightarrow \Sigma^*$ is recursively defined for any $w \in \Sigma^*$ and $a \in \Sigma$ by: $twist(aw) := a \cdot twist(w^{rev})$, and $twist(\lambda) := \lambda$.

For sets of strings L and families of languages $\mathcal{L}$ the operation *twist* is generalized as usual: $twist(L) := \{twist(w) \mid w \in L\}$ and $twist(\mathcal{L}) := \{twist(L) \mid L \in \mathcal{L}\}$.

$\square$

We see that $twist(w) = x_1 x_n x_2 x_{n-1} \cdots x_{\lfloor \frac{n}{2} \rfloor + 1}$ for any string $w \in \Sigma^*$, $w := x_1 x_2 \cdots x_{n-1} x_n$ where $x_i \in \Sigma$ for all $i \in \{1, \cdots, n\}$.

Viewed as the permutation π_{twist} of the n subscripts $1, 2, \ldots, n$, i.e. the positions of the symbols that form the string $w := x_1 x_2 \cdots x_{n-1} x_n$ this yields

$$\pi_{twist}(i) := \begin{cases} 2 \cdot i - 1 & \text{, if } 0 \leq i \leq \lceil \frac{n}{2} \rceil \\ 2(n+1-i) & \text{, otherwise} \end{cases} .$$

Twisting a context-free language obviously yields a context-sensitive language. We have $twist(\mathcal{C}f) \subsetneq \mathcal{C}s$ and the inclusion must be proper since $twist(L)$ has a semilinear Parikh image whenever L has this property. Note that $twist(L)$ may not be context-free even for a linear context-free language $L := L_{lin}$ or a one-counter language $L := L_{count}$. It is easily verified that $twist(L_{lin}) \notin \mathcal{C}f$ and $twist(L_{count}) \notin \mathcal{C}f$ for $L_{lin} := \{a^{3m} b^n c^n d^m \mid n, m \in I\!N\}$ and $L_{count} := \{a^{3m} b^m c^n d^n \mid n, m \in I\!N\}$. One verifies $twist(C) \cap \{(ad)^i (ac)^j (ab)^k \mid i, j, k \in I\!N\} = \{(ad)^m (ac)^m (ab)^m \mid m \in I\!N\}$ for $C \in \{L_{lin}, L_{count}\}$.

In order to use the operation $\frac{1}{2}$ in connection with *twist* we shall define a slightly generalzied version of this operation, compare [HoUl 79]:

2.5 Definition

For any string $w := x_1 x_2 \cdots x_n, x_i \in \Sigma$, let $\frac{1}{2}(w) := x_1 x_2 \cdots x_{\lceil \frac{n}{2} \rceil}$.

$\square$

Hence, $\frac{1}{2}(abaab) = \frac{1}{2}(abaabb) = aba$.

2.1 Lemma

Any trio which is closed with respect to *twist* is also closed under reversal and $\frac{1}{2}$.

Proof: Let $L \subseteq \Sigma^*, \$ \notin \Sigma$ be a new symbol and $f : (\Sigma \cup \{\$\})^* \longrightarrow \Sigma^*$ a homomorphism defined by $f(\$) := \lambda$ and $\forall x \in \Sigma : f(x) := x$. Then $L^{rev} = g^{-1}(twist(f^{-1}(L) \cap \{\$\}^*\Sigma^*))$ where $g : \Sigma^* \longrightarrow (\Sigma \cup \{\$\})^*$ is a homomorphism given by $\forall x \in \Sigma : g(x) := \$x$. Thus any *twist*-closed trio $\mathcal{M}(\mathcal{L})$ is closed with respect to reversal.

To express the operation $\frac{1}{2}$ by trio operations and *twist* that works for strings of both even and odd length we have to insert a dummy symbol for those of odd length and then mark half of the symbols. To do this we use an inverse homomorphism h_1^{-1}. By intersection with a suitable regular set we then can fix the position of the dummy symbol and the marked symbols.

In detail we define: $\overline{\Sigma} := \{\overline{x} \mid x \in \Sigma\}$ as a disjoint copy of Σ and the homomorphism $h_1 : (\Sigma \cup \overline{\Sigma} \cup \{\$\})^* \longrightarrow \Sigma^*$ by: $h_1(x) := h_1(\overline{x}) := x$ for all $x \in \Sigma$ and $h_1(\$) := \lambda$. Now, for any string $w \in \Sigma^*$, $h_1^{-1}(w)$ may contain an arbitrary number of extra \$-symbols and likewise barred symbols from $\overline{\Sigma}$ at any position. Then $K_1 := h_1^{-1}(L) \cap \Sigma^*\{\$, \lambda\}\overline{\Sigma}^*$ contains at most on extra symbol \$ and all and only the barred symbols at the right hand side. Define new alphabets $\Gamma := \{\langle x, y\rangle \mid x \in \Sigma, y \in \overline{\Sigma}\}$, $\Gamma_\$:= \{\langle x, \$\rangle \mid x \in \Sigma\}$ and a homomorphism $h_2 : (\Gamma_\$ \cup \Gamma)^* \longrightarrow (\Sigma \cup \overline{\Sigma} \cup \{\$\})^*$ by $h_2(\langle x, y\rangle) := xy$. Now $K_2 := h_2^{-1}(twist(K_1)) \cap (\Gamma^* \cup \Gamma^*\Gamma_\$)$ is a set of strings, each of which describes the *twist* of a string from K_1 in the projection of both components of the new symbols from $\Gamma \cup \Gamma_\$$. Since the first $\lceil\frac{|w|}{2}\rceil$ symbols of the original string w are put into the first component of the corresponding string from K_2 a simple coding will retrieve the string $\frac{1}{2}(w)$. With $h_3 : (\Gamma \cup \Gamma_\$) \longrightarrow \Sigma$ defined by $h_3(\langle x, y\rangle) := x$ one obtains $\frac{1}{2}(L) := h_3(K_2)$. The only operations we used to define $\frac{1}{2}(L)$ were trio operations and *twist* so that the lemma was proved completely.

$\square$

2.2 Lemma

Each *twist*-closed trio $\mathcal{L}$ that is in addition closed under product is also closed w.r.t. intersection.

Proof: Let $L_1, L_2 \subseteq \Sigma^*, L_1, L_2 \in \mathcal{L}$ and Γ a copy of Σ with $h : \Sigma \longrightarrow \Gamma$ being the bijection between the alphabets. By Lemma 1 ${L_2}^{rev} \in \mathcal{L}$ and then also $L_3 := g^{-1}(twist(L_1 \cdot h({L_2}^{rev})) \in \mathcal{L}$ where $g : \Sigma^* \longrightarrow (\Sigma\Gamma)^*$ is defined by $g(x) = xh(x)$ for all $x \in \Sigma$. Obviously $L_3 = L_1 \cap L_2$, and this proves the lemma.

$\square$

There exist families of languages that are closed with respect to the operations *twist* and *product* but not under intersection! The family $\mathcal{L}_{slip}$ of languages having a semi-linear Parikh image, i.e. are letter equivalent to regular sets, is such a family. This is because this family is not a trio since it is not even closed with respect to intersection by regular sets! To see this, consider the language $L := \{ab^{2^n} \mid n \in I\!N\} \cup \{b\}^*\{a\}^* \in \mathcal{L}_{slip}$, where one has $L \cap \{a\}\{b\}^* \notin \mathcal{L}_{slip}$.

This observation indicates that it might not be easy to express the operation *twist* by means of known operations in abstract formal language theory.

Using simple and standard techniques we can show that the languages *MIR*, *COPY* and their deterministic variants all are generators of the same *twist*-closed trio $\mathcal{M}_{twist}(MIR)$.

2.3 Theorem

$\mathcal{M}_{twist}(dCOPY \cup \{\lambda\}) = \mathcal{M}_{twist}(COPY) = \mathcal{M}_{twist}(MIR) =$
$\mathcal{M}_{twist}(PAL) = \mathcal{M}_{twist}(dMIR \cup \{\lambda\})$.

Proof:

(a) $COPY \in \mathcal{M}_{twist}(dCOPY \cup \{\lambda\})$ follows since *COPY* is obtained from *dCOPY* by limited erasing of the symbol c and it is well known that every trio is closed w.r.t. this operation.

(b) $MIR \in \mathcal{M}_{twist}(COPY)$ follows by observing that $MIR = twist(COPY)$. This can be shown by induction on the length and structure of the strings involved.

(c) $dMIR \in \mathcal{M}_{twist}(MIR)$, **(d)** $PAL \in \mathcal{M}_{twist}(dMIR \cup \{\lambda\})$, and **(e)** $MIR \in \mathcal{M}_{twist}(PAL)$ follow from the well known: $\mathcal{M}(dMIR \cup \{\lambda\}) = \mathcal{M}(MIR) = \mathcal{M}(PAL)$.

(f) $dCOPY \in \mathcal{M}_{twist}(dMIR)$: $K_2 := \{w\$^i w^{rev} ¢^j \mid w \in \{a,b\}^*, i,j \in I\!N \setminus \{0\}\} \in \mathcal{M}(dMIR)$ is easily proved. Likewise, $K_3 := twist(K_2) \cap (\{a,b\}\{¢\})^* \cdot \{\$¢\} \cdot (\{\$\}\{a,b\})^* \in \mathcal{M}_{twist}(dMIR)$ and then $dCOPY = f(h^{-1}(K_3))$ follows with $h : \{a,b,c,\overline{a},\overline{b}\}^* \longrightarrow \{a,b,\$,¢\}^*$ defined by $h(a) := a¢, h(b) := b¢, h(c) := \$¢, h(\overline{a}) := \$a, h(\overline{b}) := \$b$, and $f(a) := f(\overline{a}) := a, f(b) := f(\overline{b}) := b, f(c) := c$. Consequently, $dCOPY \in \mathcal{M}_{twist}(dMIR)$.

$\square$

Since the mapping *twist* only performs a permutation of the symbols that form a string it is easily seen that $\mathcal{R}e$, $\mathcal{R}ec$, and $\mathcal{C}s$ are *twist*-closed families. The family $\mathcal{M}_{\cap}(MIR)$ of quasi-realtime multipushdown languages [BoNP 74, BoGr 78] will be shown to be another *twist*-closed family.

2.4 Lemma

The family $\mathcal{M}_{\cap}(MIR)$ is closed with respect to the operation *twist*.

Proof: Let $L \in \mathcal{M}_{\cap}(MIR)$ be accepted by some nondeterministic on-line reversal-bounded multipushdown machine M_L which operates in such a way that in every computation each pushdown makes at most one reversal and runs in linear time, see [BoNP 74]. In order to accept $K := twist(L)$ we use machine M_L and add one further pushdown store to obtain machine M_K that accepts K as follows: M_K reads the symbols at odd positions of an input string $w \in K$, beginning with the first symbol of w and behaves on them exactly as the machine M_L. Beginning with the second symbol of w the symbols at even positions alternatively are pushed onto the new pushdown. After having read the last symbol of the input string the symbols from the pushdown are popped and now treated as input for the machine M_L. M_K accepts if the new pusdown is emptied and M_L accepts its input $\mathit{twist}^{-1}(w)$. Hence, M_K accepts $\mathit{twist}(L)$ and operates on each pushdown with only one reversal. It must be observed that M_K works in linear but not in quasi-realtime. That this is not a loss follows from a result in [BoNP 74] stating that the class $\mathcal{M}_\cap(\mathit{dMIR} \cap \{\lambda\})$ is closed with respect to linear erasing homomorphisms.

$\square$

Lemma 4 showed $\mathcal{M}_{\mathit{twist}}(\mathit{dMIR} \cup \{\lambda\}) \subseteq \mathcal{M}_\cap(\mathit{dMIR} \cup \{\lambda\})$ and by the following results we will be able to prove equality of these two classes. Since the family $\mathit{lin}\mathcal{C}\mathit{f}$= $\mathcal{M}(\mathit{dMIR} \cup \{\lambda\})$ =$\mathcal{M}(\mathit{MIR})$ is not closed w. r. t. product it will not simply follow from Lemma 2 that $\mathcal{M}_{\mathit{twist}}(\mathit{dMIR} \cup \{\lambda\})$ is indeed *intersection*-closed.

2.5 Lemma

$\overline{\mathit{dMIR}} \cdot \mathit{dMIR} \in \mathcal{M}_{\mathit{twist}}(\mathit{dMIR} \cup \{\lambda\})$ for $\overline{\mathit{dMIR}} := \{\overline{w} \mid w \in \mathit{dMIR}\}$.

Proof: Let $L_1 := \{c_1^{k_1}\, w\, c_2^{k_2}\, v\, c_3\, v^{rev}\, c_4^{k_4}\, w^{rev}\, c_5 \mid w \in \{a,b\}^*, v \in \{\overline{a},\overline{b}\}^*, k_1,k_2,k_4 \in I\!N \setminus \{0\}\} \in \mathit{lin}\mathcal{C}\mathit{f} = \mathcal{M}(\mathit{MIR}) \subset \mathcal{M}_{\mathit{twist}}(\mathit{MIR})$. Then let $L_2 \in \mathcal{M}_{\mathit{twist}}(\mathit{MIR})$ be defined by $L_2 := \mathit{twist}(L_1) \cap R_1$, where $R_1 := \{c_1c_5\} \cdot (\{c_1\}\{a,b\})^* \cdot \{c_1c_4\} \cdot (\{a,b\}\{c_4\})^* \cdot \{c_2c_4\} \cdot (\{c_2\}\{\overline{a},\overline{b},c_3\})^*\{c_2c_4\}$. One observes $L_2 = \{c_1c_5c_1w_1c_1w_2 \cdots c_1w_nc_1c_4w_1c_4w_2 \cdots c_4w_nc_4c_2 \cdot c_4c_2v_1c_2 \cdots v_m c_2c_3c_2v_mc_2v_{m-1} \cdots c_2v_1 \mid w_i \in \{a,b\}$ and $v_j \in \{\overline{a},\overline{b}\}\}$ and from this $L_3 := \Big\{\$_1w\$_2w\$_3v\$_4v^{rev}\$_5 \mid w \in \{a,b\}^*$ and $v \in \{\overline{a},\overline{b}\}^*\Big\} \in \mathcal{M}_{\mathit{twist}}(\mathit{dMIR} \cup \{\lambda\})$ follows easily.
By a similar technique we will finally get the stated result:
Let $L_4 := \{\$_1^{k_1}w\$_2^{k_2}w\$_3^{k_3}v\$_4v^{rev}\$_5 \mid w \in \{a,b\}^*$ and $v \in \{\overline{a},\overline{b}\}^*, k_1,k_2,k_3 \in I\!N \setminus \{0\}\} \in \mathcal{M}(L_3)$ and $L_5 := \mathit{twist}(L_4) \cap R_2$, where $R_2 := \{c_1c_5\} \cdot (\{c_1\}\{\overline{a},\overline{b},c_4\})^* \cdot \{c_1c_3\} \cdot (\{a,b\}\{c_3\})^* \cdot \{c_2c_3\} \cdot (\{c_2\}\{a,b\})^* \cdot \{c_2c_3\}$. One gets $L_6 := \{\$_1v\$_2v^{rev}\$_3w\$_4w^{rev}\$_5 \mid v \in \{\overline{a},\overline{b}\}^*$ and $w \in \{a,b\}^*\} \in \mathcal{M}(L_5)$ and finally $\overline{\mathit{dMIR}} \cdot \mathit{dMIR} \in \mathcal{M}(L_6) \subseteq \mathcal{M}_{\mathit{twist}}(\mathit{dMIR} \cup \{\lambda\}) = \mathcal{M}_{\mathit{twist}}(\mathit{MIR})$.

$\square$

Now, given two languages $L_1, L_2 \in \mathcal{M}_{\mathit{twist}}(\mathit{dMIR} \cup \{\lambda\})$ we know that each of them is obtained by finitely many applications of a-*transducer* mappings (each represented by trio operations) and the operation *twist* in any order. Our goal is to show, that each of this sequences *op₁* and *op₂* that are applied to the generator *dMIR* can be replaced by one sequence of operations which simulates these sequences on each single component of the generator $\overline{\mathit{dMIR}} \cdot \mathit{dMIR} \subset \{\overline{a},\overline{b}\}^* \cdot \{a,b\}^*$. Since this is easy as long as only a-*transducer* mappings are used we have to show that also *twist* can be applied separately to each single component. This will be proved in Lemma 6 below.

2.6 Lemma

Let $L \in \mathcal{M}_{\mathit{twist}}(\mathcal{L})$ such that $L := L_1 \cdot L_2$ for $L_1 \subseteq \Sigma_1^*, L_2 \subseteq \Sigma_2^*$ with $\Sigma_1 \cap \Sigma_2 = \emptyset$. Then

$K_1 \cdot K_2 \in \mathcal{M}_{twist}(\mathcal{L})$ for each choice of $K_i \in \{L_i, L_i^{rev}, twist(L_i)\}, i \in \{1,2\}$.

Proof: With $L_1 \cdot L_2 \in \mathcal{M}_{twist}(\mathcal{L})$ also $L_3 := \{c_1\}^* L_1 \{c_2\}^* L_2 \in \mathcal{M}_{twist}(\mathcal{L})$. Then $L_2^{rev} \cdot L_1 = f^{-1}(twist(L_3) \cap (c_1\Sigma_2)^*(\Sigma_1 c_2)^*)$ for $f(x) := \begin{cases} c_1 x\,, & \text{if } x \in \Sigma_2 \\ x c_2\,, & \text{if } x \in \Sigma_1 \end{cases}$. Now, by Lemma 1 also
(a)$L_1^{rev} \cdot L_2 = (L_2^{rev} \cdot L_1)^{rev} \in \mathcal{M}_{twist}(\mathcal{L})$.

Starting with $L_4 := L_1\{c_2\}^* L_2 \{c_3\}^* \in \mathcal{M}_{twist}(\mathcal{L})$ one easily shows
(b)$L_1 \cdot L_2^{rev} \in \mathcal{M}_{twist}(\mathcal{L})$ by the same technique.

$L_5 := twist(L_1 \cdot L_2 \cdot \{c_3\}^*) \cap (\Sigma_1 c_3)^* \Sigma_2^*$ is the basis for proving
(c)$L_1 \cdot twist(L_2) \in \mathcal{M}_{twist}(\mathcal{L})$.

Starting with $L_1 \cdot L_2 \in \mathcal{M}_{twist}(\mathcal{L})$ we find $L_2^{rev} \cdot L_1^{rev} \in \mathcal{M}_{twist}(\mathcal{L})$ since a *twist*-closed trio is closed under reversal. Applying **(b)** yields $L_2^{rev} \cdot L_1 \in \mathcal{M}_{twist}(\mathcal{L})$, **(c)** gives $L_2^{rev} \cdot twist(L_1) \in \mathcal{M}_{twist}(\mathcal{L})$, and a reversal followed by application of **(a)** finally yields
(d)$twist(L_1) \cdot L_2 \in \mathcal{M}_{twist}(\mathcal{L})$.

All other combinations stated in the lemma are now obtainable from combinations of cases **(a)** to **(d)**.

$\square$

As described in the motivation before Lemma 6 we can now apply any finite sequence of applications of trio operations (a-*transducer* mappings) and/or *twist* to the two components specified by the two generators from $\mathcal{L}$ for a language $L = L_1 \cdot L_2 \in \mathcal{M}_{twist}(\mathcal{L})$ in order to verify Theorem 7 with the help of Lemma 6:

2.7 Theorem

If $\mathcal{L}$ is closed with respect to product then $\mathcal{M}_{twist}(\mathcal{L})$ is closed under product, too.

Since we know $\overline{dMIR} \cdot dMIR \in \mathcal{M}_{twist}(dMIR \cup \{\lambda\}) = \mathcal{M}_{twist}(MIR)$ by Lemma 5 we obtain immediately:

2.8 Corollary

The family $\mathcal{M}_{twist}(MIR)$ is closed with respect to product.

Combining Lemma 2, Corollary 8, and Lemma 4 we get the main result:

2.9 Theorem

$\mathcal{M}_{twist}(MIR) = \mathcal{M}_{\cap}(MIR)$

A consequence of this new characterization of the family of languages accepted in quasi-realtime by reversal-bounded on-line multipushdown machines we find a new characterization of the recursively enumerable languages:

2.10 Corollary

$\mathcal{R}e = \hat{\mathcal{M}}_{twist}(MIR)$

In what follows we want to show how to homomorphically represent the recursively enumerable languages by twisting linear context-free languages. It is well known and easy to show that any intersection-closed trio is closed w.r.t product but closure under *twist* generally depends on the generator of this trio. It is known that each *intersection*-closed trio is also closed with respect to *shuffle*.

The proof of this theorem is similar to the one of Theorem 1 in [BaBo 74] and can more easily be described by using the operation of literal shuffle.

2.11 Theorem

A language L is recursively enumerable if and only if there exists a linear context-free language $K \in lin\mathcal{C}f$ and homomorphisms f and g such that $L = g(f^{-1}(twist(K)))$.

Proof: Let M be a deterministic Turing machine with state set Z accepting $L \subseteq \Sigma^*$ without loss of generality in such a way that all and only the halting configurations are the accepting ones. Each configuration will be represented by an instantenous description (ID) of the form uqv, where uv is the current string over the tape alphabet of M, $q \in Z$ is a state of M, and q's position in uqv indicates that M is in state q while reading the left-most symbol of v. Initial IDs are strings q_0w, where q_0 is the initial state of M and $w \in \Sigma^*$ is the input of the Turing machine.

Let K be the set of strings of the form $ID_0\$ID_1\$\cdots\$ID_{k-1}\$ID_k\$¢\,ID_k'^{rev}¢ID_{k-1}'^{rev}¢\cdots¢ID_1'^{rev}¢ID_0''^{rev}$, where $\$$ and ¢ are distinguished symbols used as markers, $ID_0'' := h(ID_0)$ encodes the initial configuration ID_0 of the TM M and uses a different copy $\Gamma := h(\Sigma)$ of the alphabet Σ which is used in all the other IDs. This is because all but the initial ID finally have to be deleted. The coding h will act as the identity on the set Z of states. ID_k' describes a final configuration of M. Also, for $1 \leq j \leq k$ no ID_j needs to be identical to ID_j', but if for some $i \in \{0, \ldots, k-1\}$ $ID_i\$$ is a substring of the set K then $¢ID_{i+1}'^{rev}$ is the corresponding substring of K if and only if ID_{i+1}' represents the configuration of M reached in one step from that represented by ID_i. That K is in fact a linear context-free language is easily verified, since the set of all strings of the form $ID_i\,\$¢\,ID_{i+1}'^{rev}$ for which $ID_i \vdash_M ID_{i+1}'$ holds clearly is an element of $lin\mathcal{C}f$. The iterated substitution of this language between the $\$$ and ¢ symbol in the middle obviously will be linear context-free again. The set of the descriptions of initial instantenous descriptions $ID_0''^{rev}$ is regular as is the set of IDs of the form ID_k in the middle. Hence, $K \in lin\mathcal{C}f$ follows.

Now, $twist(K)$ contains, among others, strings of the form $(ID_0 \sqcup\!\sqcup_{lit} ID_0'')\$¢$ $(ID_1 \sqcup\!\sqcup_{lit} ID_1')\$¢(ID_2 \sqcup\!\sqcup_{lit} ID_2')\$¢ \cdots \$¢(ID_k \sqcup\!\sqcup_{lit} ID_k')\$¢$. It will be guaranteed by applying an inverse homomorphism, that only those strings from $twist(K)$ will be taken for which $|ID_0| = |ID''_0|, h(ID_0) = ID''_0$, and $\forall i \in \{1, \dots, k\} : |ID_i| = |ID'_i|, ID_i = ID'_i$.

Let $f : (\Sigma \cup \Gamma \cup Z \cup \{\#\})^* \longrightarrow (\Sigma \cup \Gamma \cup Z \cup \{\$, ¢\})^*$ be defined by:

$$f(x) := \begin{cases} yx & , \text{ if } x = h(y) \in \Gamma \\ xx & , \text{ if } x \in \Sigma \cup Z \\ \$¢ & , \text{ if } x = \# \end{cases} .$$

Then $f^{-1}(twist(K)) =$

$$\left\{ h(ID_0)\#ID_1\# \cdots \#ID_k\# \;\middle|\; \begin{array}{l} ID_0 \text{ is an initial } ID \text{ of } M, \\ \forall i \in \{1, \dots, k\} : ID_{i-1} \vdash_M ID_i, \\ \text{and } ID_k \text{ is an acepting } ID. \end{array} \right\}.$$

Now, let $g : (\Sigma \cup \Gamma \cup Z \cup \{\#\})^* \longrightarrow \Sigma^*$ be a homomorphism that erases the symbols from the set $\Sigma \cup Z \cup \{\#\}$ and acts as h^{-1} on the set $\Gamma = h(\Sigma)$. Then, finally $g(f^{-1}(twist(K))) = L = L(M)$, since g extracts the input string of the Turing machine M from its initial configuration encoded by the prefix $h(ID_0)$ in $f^{-1}(twist(K))$.

□

Corollary 10 which is similar to the characterizations presented in [BaBo 74], [Salo 81] and [Enge 96] can now be obtained from Theorem 11 by a more direct construction instead of using the new characterization of the family $\mathcal{M}_\cap(MIR)$.

In [JaPe 94] it was proved that the family $\mathcal{M}_\cap(D_1) = PBLIND(n)$ is closed with respect to *twist*. We will now show that for each $k \geq 1$ the family $\mathcal{M}(C_k)$ of languages accepted by blind k-counter machines in realtime is *twist*-closed, too. To do this let us recall that each blind k-counter machine M can easily be described by a finite state transition diagram in which a directed arc from state p_1 to p_2 is inscribed by the input symbol x to be processed and a vector $\Delta \in \{+1, 0, -1\}^k$ used for updating the counters by adding Δ to the current contents $C_1 \in \mathbb{Z}^k$ of the counters . This will be written as $p_1 \xrightarrow[\Delta]{x} p_2$. A string $w = x_1x_2x_3 \cdots x_{n-1}x_n,\ x_i \in \Sigma$ is accepted by M, iff there exists a path in the transition diagram of the form $q_0 \xrightarrow[\Delta_1]{x_1} q_1 \xrightarrow[\Delta_2]{x_2} q_2 \xrightarrow[\Delta_3]{x_3} q_3 \cdots q_{n-1} \xrightarrow[\Delta_n]{x_n} q_n$, where q_0 (q_n) is an initial (resp. final) state and $\Sigma_{i=1}^{n}\Delta_i = 0$ in each component. The machine starts with empty counters and accepts only when all counters are zero again. It is easy to construct a blind k-counter machine M_2 that accepts L^{rev} from the machine M_1 that accepts $L \in \mathcal{M}(C_k)$: One just has to revert the arcs in the state transition diagram of M_1 and exchange the sets of final and initial states. Now it is not difficult to show that $\mathcal{M}(C_k)$ is *twist*-closed for each $k \geq 1$.

2.12 Theorem

$\forall k \in \mathbb{N}, k \geq 1 : \mathcal{M}_{twist}(C_k) = \mathcal{M}(C_k)$.

Proof: Let $L \in \mathcal{M}(C_k), L \subseteq \Sigma^*$, then there exists a blind k-counter machine M which accepts $L = L(M)$ in realtime. In order to accept the set $twist(L_e)$ we modify the machine

M to a new machine M_{twist} as follows: Let Q (Q_0, and Q_f) be the set of states (initial and final states, resp.) of the machine M, then $Q_{twist} := Q^2 \times \{o, e\}$ is the set of states of M_{twist}. The sets of initial (and final) states $Q_{0,twist}$ ($Q_{f,twist}$, resp.) of M_{twist} are given by $Q_{0,twist} := \{(p_0, p_f, o) \mid p_0 \in Q_0, p_f \in Q_f\}$ and $M_{f,twist} := \{(p, p, o), (p, p, e) \mid p \in Q\}$. Let $w \in L$, $w = x_1x_2x_3 \cdots x_{n-1}x_n$, $x_i \in \Sigma$ be a string of length $|w| = n$ which is accepted by M in a sequence of transitions $q_0 \xrightarrow[\Delta_1]{x_1} q_1 \xrightarrow[\Delta_2]{x_2} q_2 \xrightarrow[\Delta_3]{x_3} q_3 \cdots q_{n-1} \xrightarrow[\Delta_n]{x_n} q_n$. The new machine M_{twist} now uses the finite control of M in the first components of the elements in Q_{twist} in each odd step beginning with the first move, while it is used every second (even) step in the second components backwards. The sequence of transitions of M_{twist} accepting $twist(w)$ now would be

$$\begin{pmatrix} q_0 \\ q_n \\ o \end{pmatrix} \xrightarrow[\Delta_1]{x_1} \begin{pmatrix} q_1 \\ q_n \\ e \end{pmatrix} \xrightarrow[\Delta_n]{x_n} \begin{pmatrix} q_1 \\ q_{n-1} \\ o \end{pmatrix} \xrightarrow[\Delta_2]{x_2} \begin{pmatrix} q_2 \\ q_{n-1} \\ e \end{pmatrix} \xrightarrow[\Delta_{n-1}]{x_{n-1}} \cdots \xrightarrow[\Delta_{\lfloor \frac{n}{2} \rfloor +1}]{x_{\lfloor \frac{n}{2} \rfloor +1}} \begin{pmatrix} q_{\lceil \frac{n}{2} \rceil} \\ q_{\lceil \frac{n}{2} \rceil} \\ x \end{pmatrix},$$

where $x \in \{o, e\}$ depends on the length of the input string:

The last step in this computation is $\begin{pmatrix} q_{\frac{n}{2}} \\ q_{\frac{n}{2}+1} \\ e \end{pmatrix} \xrightarrow[\Delta_{\frac{n}{2}+1}]{x_{\frac{n}{2}+1}} \begin{pmatrix} q_{\frac{n}{2}} \\ q_{\frac{n}{2}} \\ o \end{pmatrix}$, if n is even, and is $\begin{pmatrix} q_{\lfloor \frac{n}{2} \rfloor} \\ q_{\lceil \frac{n}{2} \rceil} \\ o \end{pmatrix} \xrightarrow[\Delta_{\lceil \frac{n}{2} \rceil}]{x_{\lceil \frac{n}{2} \rceil}} \begin{pmatrix} q_{\lceil \frac{n}{2} \rceil} \\ q_{\lceil \frac{n}{2} \rceil} \\ e \end{pmatrix}$, otherwise.

Conversely, every accepting computation in M_{twist} can be unfolded to yield a valid computation in M showing that only strings of the form $twist(w)$, $w \in L(M)$ are accepted by M_{twist}.

$\square$

2.13 Corollary

$\forall k \in I\!N, k \geq 1 : \mathcal{M}_{twist}(B_k) = \mathcal{M}(C_k)$.

Proof: From Lemma 12 and $B_k \in \mathcal{M}(C_k)$ we see $\mathcal{M}_{twist}(B_k) \subseteq \mathcal{M}(C_k)$. To show the converse we have to verify $C_k \in \mathcal{M}_{twist}(B_k)$. Let $¢ \notin \Gamma_k$ and the homorphism $h_{¢}$: $(\Gamma_k \cup \{¢\})^* \longrightarrow \Gamma_k^*$ be defined by $h(x) := x$, if $x \in \Gamma_k$ and $h(¢) := \lambda$. Then $L_k := twist(h^{-1}(B_k)) \cap \{¢b_i, a_j¢ \mid b_i, a_j \in \Gamma_k\}^*$ is an element of $\mathcal{M}_{twist}(B_k)$ and the pairs $¢b_i, a_j¢$ may appear in any order for all $1 \leq i, j \leq k$. Hence, applying the inverse homomorphism $g : \Gamma_k^* \longrightarrow (\Gamma_k \cup \{¢\})^*$ with $g(a_i) := a_i¢$ and $g(b_i) := ¢b_i$ we see $C_k = g^{-1}(L_k) \in \mathcal{M}_{twist}(B_k)$.

$\square$

Greibach showed $C_1 \in \mathcal{M}(B_3)$ in [Grei 78]. We want to show that it is sufficient to use only one more counter to accept C_k using only k+1 partially blind reversal-bounded counters. This should also be compared with Theorem 1 in [Hrom 85].

2.14 Theorem

$\forall k \in I\!N, k \geq 1 : \mathcal{M}(C_k) \subsetneq \mathcal{M}(B_{k+1})$.

Proof: "$\subseteq$". Let C_k be accepted by some blind k-counter machine working in realtime. The new reversal-bounded (k+1)-counter machine M_{k+1} is given as follows: One counter, call it z_0, is used to non-deterministically find the middle of each string $w \in C_k$ by adding 1 in each step when reading a prefix u of $w = uv$. We call this the first phase of the work of M_{k+1}. Then, non-deterministically this phase is stopped and in the following, second, phase the counter z_0 is decreased by 1 in each and every step. One has $|u| = |v| = \frac{|w|}{2}$ if and only if this counter reached zero after reading the last symbol of w. All other counters z_1 to z_k are treated differently in the first and the second phase of M_{k+1}'s computation: If $\Delta = (\delta_1, \ldots, \delta_k)$ is a counter-update used in the first phase of M_k, then $\Delta' := (\delta_1 + 1, \ldots, \delta_k + 1) \in \{2, 1, 0\}^k$ is the non-decreasing counter-update in M_{k+1} to be used instead. Likewise, if $\Delta = (\delta_1, \ldots, \delta_k)$ is a counter-update used in the second phase of M_k, then $\Delta' := (\delta_1 - 1, \ldots, \delta_k - 1) \in \{-2, -1, 0\}^k$ is the non-increasing counter-update to be used in M_{k+1} instead. Since the first and the second phase consist of equally many steps, the overall change of the counters is zero again, and exactly the strings form C_k are accepted using k+1 reversal-bounded partially blind counters. Since the new counter-updates now are elements of $\{-2, -1, 0, 1, 2\}^{k+1}$ and not of $\{-1, 0, 1\}^{k+1}$ we have to modify the machine M_{k+1} by splitting moves that increase (or decrease) a counter by 2 into two moves that increase (or decrease) this counter by 1 and all other counters are treated as before in the first step and stay stationary in the second step. This modification gives a partially blind (k+1)-counter machine that accepts C_k in quasi-realtime and with one reversal on each of its counters, hence $\mathcal{M}(C_k) \subseteq \mathcal{M}(B_{k+1})$.

In order to see the strictness of this inclusion we use known results from Ginsburg, [Gins 75], Greibach, [Grei 76], or Latteux, [Latt 78, Latt 79], where it was shown that $B_{k+1} \cap \prod_{i=1}^{k+1} \{a_i\}^*\{b_i\}^*$ is not an element of $\mathcal{M}(C_k)$, hence $\mathcal{M}(C_k) \subsetneq \mathcal{M}(B_{k+1})$ for each $k \geq 1$.

$\square$

Consequently, $\bigcup_{i \geq 0} \mathcal{M}(C_i) = \mathcal{M}_\cap(C_1)$ forms a strict hierarchy of *twist*-closed semiAFLs and therefore cannot be principal as *twist*-closed semiAFL.

The language $E_k := \{a_1^n a_2^n \cdots a_k^n \mid n \in I\!N\}$ was used to separate the classes $\mathcal{M}(C_k) \subsetneq \mathcal{M}(C_{k+1})$, see [Latt 77, Latt 78, Latt 79], by proving $E_{2k+1} \notin \mathcal{M}(C_k)$. Ginsburg, [Gins 75] Example 4.5.2, has shown $\mathcal{M}(B_k) \subsetneq \mathcal{M}(B_{k+1})$ and in [Grei 76, Jant 79a] it was shown that $B_{k+1} \notin \mathcal{M}(D_k)$. Kortelainen has shown in [Kort 80] Theorem 2.2.2, that $E_{k+1} \notin \mathcal{F}(COM_k)$, where COM_k denotes the class of commutative languages over an alphabet of k symbols. Note, $\mathcal{M}(C_k) \subsetneq \mathcal{M}(COM_{2k})$.

We conjecture the following sharpening of the above results: $\mathcal{M}(B_k) \subsetneq \mathcal{M}(C_k)$, for all $k \in I\!N, k \geq 1$.

$\mathcal{M}(B_1) \subsetneq \mathcal{M}(C_1)$ can be shown easily: $\mathcal{M}(B_1) \subseteq \mathcal{M}(LCF)$ but $\mathcal{M}(C_1) \not\subseteq \mathcal{M}(LCF)$, and only $\mathcal{M}(B_k) \subseteq \mathcal{M}(C_k)$ is obvious.

Acknowledgement: I thank Julia Maas for discussing various aspects of the twist operation and especially Olaf Kummer for Theorems 12 and 14.

References

[BaBo 74] B.S. Baker and R.V. Book. Reversal-bounded multipushdown machines, J. Comput. Syst. Sci., **8** (1974) 315–332.

[Bers 80] J. Berstel, Transductions and Context-free Languages, Teubner Stuttgart (1980).

[BoGr 70] R.V. Book and S. Greibach. Quasi-realtime languages, Math. Syst. Theory **19** (1970) 97–111.

[BoGr 78] R.V. Book and S. Greibach. The independence of certain operations on semi-AFLs, RAIRO Informatique Théorique, **19** (1978) 369–385.

[BoNP 74] R.V. Book, M. Nivat, and M. Paterson. Reversal-bounded acceptors and intersections of linear languages, Siam J. on Computing, **3** (1974) 283–295.

[Bran 87] F.J. Brandenburg. Representations of language families by homomorphic equality operations and generalized equality sets, Theoretical Computer Science, **55** (1987) 183–263.

[Bran 88] F.J. Brandenburg. On the intersection of stacks and queues, Theoretical Computer Science, **58** (1988) 69–80.

[Chan 81] T.-H. Chan. Reversal complexity of counter machines, in: Proc. 13th annual ACM Sympos. on Theory of Computing, Milwaukee, Wisconsin, (1981) 146–157.

[Culi 79] K. Culik. A purely homomorphic characterization of recursively enumerable sets, J. ACM **26** (1979) 345–350.

[ĎuHr 87] P. Ďuriš and J. Hromkovič. Zerotesting bounded one-way multicounter machines. Kybernetika, **23** (1) (1987) 13–18.

[EnRo 79] J. Engelfriet and G. Rozenberg. Equality languages and fixed point languages, Information and Control, **43** (1979) 20–49.

[Enge 96] J. Engelfriet. Reverse twin shuffles, Bulletin of the EATCS, vol. **60** (1996) 144.

[FiMR 68] P.C. Fischer, A.R. Meyer, and A.L. Rosenberg. Counter machines and counter languages, Math. Syst. Theory, **2** 1968 265–283.

[Gins 75] S. Ginsburg, Algebraic and Automata Theoretic Properties of Formal Languages, North Holland Publ. Comp. Amsterdam (1975).

[GiGo 71] S. Ginsburg and J. Goldstein. Intersection-closed full AFL and the recursively enumerable languages, Information and Control, **22** (1973) 201–231.

[GiGr 70] S. Ginsburg and S. Greibach, Principal AFL, J. Comput. Syst. Sci., **4** (1970) 308–338.

[Grei 76] S. Greibach. Remarks on the complexity of nondeterministic counter languages, Theoretical Computer Science, **1** (1976) 269–288.

[Grei 78] S. Greibach. Remarks on blind and partially blind one-way multicounter machines, Theoretical Computer Science, **7** (1978) 311–236.

[Hrom 85] J. Hromkovič. Reversal bounded multicounter machines. Computers and Artificial Intelligence, **4** (1985) 361–366.

[HoUl 79] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley Publ. Comp. (1997).

[Jant 79a] M. Jantzen. On the hierarchy of Petri net languages, R.A.I.R.O., Informatique Théorique, **13** (1979) 19–30.

[Jant 79b] M. Jantzen. On zerotesting-bounded multicounter machines. In: Proc. 4th GI-Conf. Lecture Notes in Compuer Science, vol. **67**, Springer, Berlin, Heidelberg, New York (1979) 158–169.

[JaPe 87] M. Jantzen and H. Petersen. Petri net languages and one-sided reductions of context-free sets, in: (K Voss, H. Genrich, and G. Rozenberg, eds.) Concurrency and Nets, Springer, Berlin, Heidelberg, New York (1987) 245–252.

[JaPe 91] M. Jantzen and H. Petersen. Twisting Petri net languages and how to obtain them by reducing linear context-free sets, in: Proc. 12th Internat. Conf. on Petri Nets, Gjern (1991) 228–236.

[JaPe 94] M. Jantzen and H. Petersen. Cancellation in context-free languages: enrichment by reduction. Theoretical Computer Science, **127** (1994) 149–170.

[Kort 80] J. Kortelainen.Properties of trios and AFLs withbounded or commutative generators, Dept. of Mathematics, Univ. of Oulu, Finland, Techn. Report No. 53 (1980).

[Kosa 82] S.R. Kosaraju. Decidability of reachability of vector addition systems, 14th Annual ACM Symp. on Theory of Computing, San Francisco, (1982) 267–281.

[Latt 77] M. Latteux. Cônes rationnels commutativement clos. R.A.I.R.O., Informatique Théorique, **11** (1977) 29–51.

[Latt 78] M. Latteux.Langages commutatifs, Thesè Sciences Mathematiques, Univ. Lille (1978).

[Latt 79] M. Latteux. Cônes rationnels commutatifs. J. Comput. Syst. Sci., **18** (3) (1979) 307–333.

[Mayr 84] E.W. Mayr. An algorithm for the general Petri net reachability problem, SIAM J. of Computing, **13** (1984) 441–459.

[Salo 78] A. Salomaa. Equality sets for homomorphisms and free monoids, Acta Cybernetica **4** (1978) 127–139.

[Salo 81] A. Salomaa. Jewels of formal Language Theory, Computer Science Press, Rockville (1981).