Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 2152–2164

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Finite turns and the regular closure of linear context-free languages

Martin Kutrib[a], Andreas Malcher[b,*]

[a] *Institut für Informatik, Universität Giessen, Arndtstr. 2, D-35392 Giessen, Germany*
[b] *Institut für Informatik, Johann Wolfgang Goethe Universität, D-60054 Frankfurt am Main, Germany*

Received 28 July 2006; received in revised form 8 May 2007; accepted 15 May 2007
Available online 2 June 2007

Abstract

Turn bounded pushdown automata with different conditions for beginning a new turn are investigated. Their relationships with closures of the linear context-free languages under regular operations are studied. For example, automata with an unbounded number of turns that have to empty their pushdown store up to the initial symbol in order to start a new turn are characterized by the regular closure of the linear languages. Automata that additionally have to re-enter the initial state are (almost) characterized by the Kleene star closure of the linear languages. For both a bounded and an unbounded number of turns, requiring to empty the pushdown store is a strictly stronger condition than requiring to re-enter the initial state. Several new language families are obtained which form a double-stranded hierarchy. Closure properties of these families under AFL operations are derived. The regular closure of the linear languages share the strong closure properties of the context-free languages, i.e., the family is a full AFL. Interestingly, three natural new language families are not closed under intersection with regular languages and inverse homomorphism. Finally, an algorithm is presented parsing languages from the new families in quadratic time.

Keywords: Finite turn pushdown automata; Computational capacity; Time-efficient recognizers; Closures of languages; Context-free languages

1. Introduction

Context-free languages are one of the most important and most developed area of formal language theory. The particular interest is probably due to their great practical importance. Since the known upper bound on the time complexity for context-free language recognition still exceeds $O(n^2)$, there is a considerable interest in language families that admit efficient recognizers, but decrease the descriptional capacity only slightly. A well-known sub-family are the deterministic context-free languages that parse rapidly in linear time. On the other hand, considering only deterministic context-free languages might be too rigid from the descriptional capacity point of view. Clearly, any ambiguity would be lost. Since deterministic context-free languages are not closed under the regular operations union, concatenation, and Kleene star, the regular closure of the deterministic context-free languages increases the descriptional capacity again. In fact, the regular closure contains, e.g., inherently ambiguous languages such as $\{a^m b^m c^n\} \cup \{a^m b^n c^n\}$ [8], while the time complexity is still of order $O(n)$ [3].

* Corresponding author. Tel.: +49 69 7982 8413; fax: +49 69 7982 8334.
E-mail addresses: kutrib@informatik.uni-giessen.de (M. Kutrib), a.malcher@em.uni-frankfurt.de (A. Malcher).

0166-218X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2007.05.021

Another well-known sub-family of the context-free languages that has an attractively efficient recognition algorithm taking $O(n^2)$ time and $O(n)$ space are the linear languages. On one hand, considering linear languages does not remove ambiguity completely, on the other hand, one may consider strict superclasses without increasing the recognition time. As for deterministic context-free languages we consider closures of linear languages which are closed under union, but are not closed under concatenation and Kleene star. The concatenation closure is included in the metalinear languages which are accepted with time complexity $O(n^2)$. The latter follows by the family of languages accepted by a certain massively parallel automata model, the so-called deterministic real-time two-way cellular automata. It is known that this family is recognized with $O(n^2)$ time and $O(n)$ space [11,16] and contains the metalinear languages [12].

Going one step ahead, the regular closure of the linear languages is reached. Considering regular expressions whose symbols are linear languages, in Ref. [17] a recognition time complexity of order $O(n^2)$ is stated. In Section 6 a parsing algorithm with time complexity $O(n^2)$ is presented which generalizes these results.

As for the deterministic context-free languages there is an automaton model for linear languages. Restricting a pushdown automaton (PDA) such that the height of its stack is only allowed to increase and then to decrease, thus performing only one turn, leads to the definition of one-turn PDAs [5]. It is known that these PDAs can grammatically be characterized by linear grammars. Again, motivated by fast recognition algorithms this notion has been introduced in [5]. PDA that are allowed to perform a fixed number of turns may reject inputs faster than general PDA, since a computation can be halted as soon as it exceeds the number of turns. These finite turn pushdown automata (tPDA) are grammatically characterized by ultralinear languages [5]. In [18] the so-called turn-bounded grammars have been defined that form another class of grammars characterizing the ultralinear languages. Moreover, it has been shown that every ε-free ultralinear language is accepted by an ε-free one-state finite tPDA.

Once there is a device with a resource that is bounded by some arbitrary but fixed constant k, the question for the computational power induced by this resource arises immediately. For finite tPDA it has been shown that the class of languages accepted by k-tPDA is strictly included in the class of languages accepted by $(k+1)$-tPDA [7]. Recently, in [14] it has been shown that the recognition of finite tPDA is logarithmically space bounded, they belong to the complexity class NL. Generalizing from one-turn to k-turn slightly differently, i.e., requiring a k-tPDA to empty its pushdown store up to the initial symbol before starting a new turn, leads to a class of automata which are grammatically characterized by metalinear grammars. For a fixed k one obtains a k-linear language. Again, an infinite hierarchy depending on k has been shown [4]. Moreover, the proper inclusion of the metalinear languages in the ultralinear languages is known [1]. So, sticking with finite turn automata, the computational capacity may depend on the requirements that have to be met before starting a new turn.

The main goal of this paper is to investigate the relationships between finite turn automata with different conditions for beginning a new turn, on one hand, and the characterizations of the accepted language families by sub-families of the context-free languages and their closures under regular operations, on the other hand.

The paper is organized as follows. In the following section we present some basic notions and definitions. In particular, we briefly recall the definitions of meta- and ultralinear languages and define tPDA and conditions that have to be met in order to start a new turn. We consider devices with no further condition, devices that have to empty their pushdown store, devices that have to re-enter the initial state, and devices that have to re-initialize completely, which means to empty their pushdown store and to re-enter the initial state. Section 3 deals with the computational power of tPDA. We will show characterizations in terms of closures of linear languages. So, in some sense we can bridge the description of language families by automata and by algebraic operations.

Interestingly, by unbounded turn automata that have to empty their pushdown store between turns, exactly the regular closure of the linear languages is characterized. The results are summarized in Table 1. In Section 4, basically, the closure properties of the families in question under the AFL-operations (union, concatenation, Kleene star, homomorphism, inverse homomorphism, and intersection with regular languages) are exhibited. It turns out that the regular closure of linear languages is the sole family that shares the properties with the context-free languages, they form a full AFL. Surprisingly, we obtain three language families that are not closed under intersection with regular languages and inverse homomorphism. One of these families is the Kleene star closure of the linear languages. The results are summarized in Table 2. In Section 5, relations between the automata classes are derived. The results reveal a double-stranded hierarchy. The section is concluded with a diagram summarizing the shown inclusions. Finally, in Section 6 we present a simple variant of the Cocke–Younger–Kasami algorithm which shows in an easy way that the regular closure of linear languages and thus all new language families discussed in this paper can be parsed in quadratic time.

2. Preliminaries

Let Σ^* denote the set of all words over the finite alphabet Σ. The empty word is denoted by ε and $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. For the length of a word w we write $|w|$. Set inclusion and strict set inclusion are denoted by $\subseteq$ and $\subset$, respectively. We assume that the reader is familiar with the common notions in formal language theory as presented in [10,15]. We write REG, CFL, and LIN for the families of regular, context-free, and linear languages, respectively.

2.1. Regular closures, meta- and ultralinear languages

Let $\mathscr{L}$ be a family of languages and $op_1, \ldots, op_k$, $k \in \mathbb{N}$, be a finite number of operations defined on $\mathscr{L}$. Then $\Gamma_{op_1,\ldots,op_k}(\mathscr{L})$ denotes the *smallest family of languages which contains all members of* $\mathscr{L}$ *and is closed under* $op_1, \ldots, op_k$. In particular, we consider the operations union ($\cup$), concatenation ($\bullet$), and Kleene star ($*$), which are called *regular operations*. Accordingly, we write Γ_{REG} for the *regular closure*, i.e., $\Gamma_{\cup,\bullet,*}$.

A *context-free grammar* $\mathscr{G} = \langle N, T, S, P\rangle$, where N and T are disjoint alphabets of non-terminals and terminals, respectively, $S \in N$ is the axiom, and P is a finite set of productions, is said to be *metalinear* if all productions of P are of the following forms $S \to A_1A_2\cdots A_m$, where $A_i \in N - \{S\}$, $A \to u$, where $A \in N - \{S\}$, and $u \in (T^*(N - \{S\})T^*) \cup T^*$. The family of languages generated by metalinear grammars is called metalinear languages and is denoted by METALIN.

The width of a metalinear grammar is $\max\{m \mid S \to A_1A_2\cdots A_m\}$. The family of languages generated by metalinear grammars of width k is called *k-linear* languages and is denoted by LIN(k). Metalinear languages of width 1 are called *linear languages*.

A context-free grammar is said to be *ultralinear* (cf. [5]) if N is a union of disjoint (possibly empty) subsets $N_0, \ldots, N_n$ with the following property: for each N_i and each $A \in N_i$, each production with left-hand side A is of the form $A \to u$, where either $u \in (T^*N_iT^*)$ or $u \in (T \cup N_0 \cup \cdots \cup N_{i-1})^*$. The family of languages generated by ultralinear grammars is called ultralinear languages and is denoted by ULTRALIN.

A context-free grammar is said to be *non-terminal bounded* (cf. [5]) if there exists an integer k with the following property: If $A \Rightarrow^* w$, $w \in (N \cup T)^*$, $A \in N$, then w has at most k occurrences of non-terminals. It is shown in [5] that a context-free grammar is ultralinear if and only if it is non-terminal bounded.

2.2. Turn pushdown automata

Considering a computation of a PDA we call a step that reverses the length of the pushdown from non-decreasing to decreasing a *turn*. We study PDA whose number of turns is bounded or unbounded. Moreover, we are interested in devices that obey several conditions to start a new turn. To be more precise we continue with the formalization of turn pushdown automata (tPDA).

A PDA is a system $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$, where Q is a finite set of states, Σ is a finite input alphabet, Γ is a finite pushdown alphabet, δ is a mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^*$ called the transition function, $q_0 \in Q$ is the initial state, $Z_0 \in \Gamma$ is a particular pushdown symbol, called the bottom-of-pushdown symbol, which initially appears on the pushdown store, and $F \subseteq Q$ is a set of accepting states.

A *configuration* of a PDA is a triple (q, w, γ), where q is the current state, w the unread part of the input, and γ the current content of the pushdown store, the leftmost symbol of γ being the top symbol. If p, q are in Q, a is in $\Sigma \cup \{\varepsilon\}$, w is in Σ^*, γ and β are in Γ^*, and Z is in Γ, then we write $(q, aw, Z\gamma) \vdash_{\mathscr{M}} (p, w, \beta\gamma)$, if the pair (p, β) is in $\delta(q, a, Z)$.

In order to simplify matters, we require that during any computation the bottom-of-pushdown symbol appears only at the bottom of the pushdown store. Formally, we require that if (p, β) is in $\delta(q, a, Z)$, then either β does not contain Z_0 or $\beta = \beta' Z_0$, where β' does not contain Z_0, and $Z = Z_0$.

As usual, the reflexive transitive closure of $\vdash_{\mathscr{M}}$ is denoted by $\vdash^*_{\mathscr{M}}$. The subscript $\mathscr{M}$ will be dropped whenever the meaning remains clear. Furthermore, the meaning of Γ will never conflict with the closure operator.

The *language accepted* by $\mathscr{M}$ with accepting states is

$$T(\mathscr{M}) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \gamma) \quad \text{for some } q \in F \text{ and } \gamma \in \Gamma^*\}.$$

Now we turn to turns. Let $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a PDA. A sequence of configurations (computation) $(q_1, w_1, \gamma_1)\vdash_{\mathscr{M}} \cdots \vdash_{\mathscr{M}} (q_k, w_k, \gamma_k)$ is called *one-turn* if there exist integers $1 < i \leqslant j < k$ such that $|\gamma_1| \leqslant \cdots \leqslant |\gamma_{i-1}| < |\gamma_i| \leqslant |\gamma_{i+1}| \leqslant \cdots |\gamma_j| > |\gamma_{j+1}| \geqslant \cdots \geqslant |\gamma_k|$.

Next we put several restrictions on PDAs and call the resulting devices tPDA.

A computation $c_0\vdash \cdots \vdash c_m$ is called *k-turn* if there are integers $0{=}i_0, \ldots, i_l{=}m$ with $l \leqslant k$ such that for $j{=}0, \ldots, l{-}1$ the subsequences $c_{i_j}\vdash \cdots \vdash c_{i_{j+1}}$ are one-turn, respectively.

In particular, we will bound the number of allowed turns by constants k, or allow an arbitrary number of turns, which is indicated by ∞. If every accepting computation of $\mathscr{M}$ is a k-turn computation, then $\mathscr{M}$ is a k-tPDA, for an unbounded number of turns we write ∞-tPDA.

The next step is to put restrictions on the configurations c_{i_j}. That is, conditions that have to be met in order to start a new turn. In particular, if all configurations c_{i_j}, $j = 0, \ldots, l-1$ are of the form: (i) (q, w, Z_0), then the device has to empty its pushdown store and is denoted by (k, Z_0)-tPDA, (ii) (q_0, w, γ), then the device has to return to the initial state q_0 and is denoted by (k, q_0)-tPDA, and (iii) (q_0, w, Z_0), then the device has to reinitialize completely, which means to empty its pushdown store and to return to the initial state, and is denoted by (k, q_0, Z_0)-tPDA. We use the prefixes (∞, Z_0), (∞, q_0), and (∞, q_0, Z_0) for an unbounded number of turns.

In general, we denote the *family of languages accepted* by devices of type X by $\mathscr{L}(X)$.

3. Power and characterizations of tPDA

This section is devoted to study the computational power of tPDAs. We will show characterizations in terms of closures of linear languages. So, in some sense we can bridge the description of language families by automata and by algebraic operations.

The following characterizations of k-linear, metalinear, and ultralinear languages are well known. The results may be found in [1,5].

Theorem 1.

(1) *For any $k \in \mathbb{N}$, a language L belongs to* LIN(k) *if and only if L is accepted by a* (k, Z_0)-tPDA.
(2) *Thus, a language L belongs to* METALIN *if and only if there is a $k \in \mathbb{N}$ such that L is accepted by a* (k, Z_0)-tPDA.
(3) *A language L belongs to* ULTRALIN *if and only if there is a $k \in \mathbb{N}$ such that L is accepted by a* k-tPDA.

So far, we have defined the acceptance by accepting states. For PDAs the equivalent acceptance by empty pushdown is well known. Accordingly, the *language accepted* by some tPDA $\mathscr{M}$ with *empty pushdown* is

$$N(\mathscr{M}) = \{w \in \Sigma^* \mid (q_0, w, Z_0)\vdash^*(q, \varepsilon, \varepsilon) \text{ for some } q \in Q\}.$$

The next lemma shows that, in fact, both acceptance modes are equivalent for tPDAs, too. The crucial point is that the proof for classical PDAs relies on the fact that a new bottom-of-pushdown symbol can be used during the simulation. This causes no problems if a new turn may be started when the pushdown store is not empty. For these cases we obtain the equivalence immediately from the proof for classical PDAs. For acceptors that have to empty their pushdown store we provide the following lemma.

Lemma 2. *Let $\mathscr{M}$ be a* tPDA *that has to empty its pushdown store in order to start a new turn. Then there exists a* tPDA $\mathscr{M}'$ *of the same type such that $T(\mathscr{M}) = N(\mathscr{M}')$.*

Proof. Let $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a tPDA that accepts by accepting states.

Let $p \notin Q$ be a new state. We define $\mathscr{M}' = \langle Q \cup \{p\}, \Sigma, \Gamma, \delta', q_0, Z_0, \emptyset\rangle$ as follows.

Basically, $\mathscr{M}'$ has the same transitions as $\mathscr{M}$. Additionally, for every transition $(q', \gamma) \in \delta(q, a, Z)$ with $q' \in F$ we add a transition $(p, \varepsilon) \in \delta(q, a, Z)$. Furthermore, for all $Z \in \Gamma$ we add $(p, \varepsilon) \in \delta(p, \varepsilon, Z)$. Finally, we remove all transitions $(q', \varepsilon) \in \delta(q, a, Z_0)$ with $q' \notin F$. This avoids that during the simulation the pushdown store gets empty, when $\mathscr{M}$ empties its pushdown store without accepting. It is easy to see that $\mathscr{M}'$ accepts $T(\mathscr{M})$ with empty pushdown store. Moreover, the modifications do not require a new turn.

Conversely, let $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a tPDA that accepts by empty pushdown store.

Let $f \notin Q$ be a new state. We define $\mathscr{M}' = \langle Q \cup \{f\}, \Sigma, \Gamma, \delta', q_0, Z_0, \{f\}\rangle$ as follows. Every transition $(q', \varepsilon) \in \delta(q, a, Z_0)$ with $q, q' \in Q$ and $a \in \Sigma \cup \{\varepsilon\}$ is changed into $(f, \varepsilon) \in \delta(q, a, Z_0)$. It is easy to see that $\mathscr{M}'$ accepts $T(\mathscr{M})$ with empty pushdown store and accepting states. Again, the modifications do not change the type of the tPDA $\mathscr{M}$. □

The next lemma shows what happens if the tPDA has to return to the initial state in order to start a new turn. Intuitively, this means that no information can be transferred from one turn to the next turn via states. But at this point we still allow information to be transferred via pushdown store.

Lemma 3. (1) *A language L is context free if and only if L is accepted by a (∞, q_0)-tPDA.*
(2) *A language L is ultralinear if and only if there is a $k \in \mathbb{N}$ such that L is accepted by a (k, q_0)-tPDA.*

Proof. It is well known that every context-free language can be accepted by a one-state PDA (cf. [10]). This shows the first assertion immediately.
In [18] a similar result has been shown for ultralinear languages and one-state k-tPDA. This shows the second assertion. □

Theorem 1 shows that k-tPDAs which have to empty their pushdown stores in order to start a new turn accept k-linear languages. So, the metalinear languages are characterized by (k, Z_0)-tPDAs, where k is an arbitrary not fixed constant. Before we turn to one of our main results, which deals with the situation in which an unbounded number of turns is allowed, we recall the result of Theorem 1 in terms of what we are particularly interested in, namely in terms of a closure of the linear languages.

Corollary 4. *A language L belongs to the union closure of the concatenation closure of linear languages $\Gamma_\cup(\Gamma_\bullet(\mathrm{LIN}))$ if and only if there is a $k \in \mathbb{N}$ such that L is accepted by a (k, Z_0)-tPDA.*

Proof. The assertion follows by $\Gamma_\cup(\Gamma_\bullet(\mathrm{LIN})) = \mathrm{METALIN}$ and Theorem 1. □

Intuitively, the devices under consideration of the next theorem cannot transfer information from one turn to the next turn via pushdown store but via states.

Theorem 5. *A language L is accepted by a (∞, Z_0)-tPDA if and only if L belongs to the regular closure of the linear languages $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$.*

Proof. Since any linear language is accepted by some (∞, Z_0)-tPDA, and the language family $\mathscr{L}((\infty, Z_0)\text{-tPDA})$ is closed under the regular operations (cf. Lemma 10), any language $L \in \Gamma_{\mathrm{REG}}(\mathrm{LIN})$ is accepted by some (∞, Z_0)-tPDA.
To show the converse let $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset\rangle$ be a (∞, Z_0)-tPDA that accepts by empty pushdown store. Roughly, the idea is as follows. We first transform $\mathscr{M}$ into an equivalent context-free grammar $\mathscr{G}$. Then we show that $\mathscr{G}$ is of a certain form which allows us to consider $\mathscr{G}$ as a right-linear grammar $\mathscr{G}'$ whose terminal symbols represent linear languages generated by non-terminals of $\mathscr{G}$. Next, $\mathscr{G}'$ is transformed into an equivalent regular expression $\mathscr{E}$, where each symbol in $\mathscr{E}$ still represents a linear language generated by non-terminals of $\mathscr{G}$. Therefore, $T(\mathscr{M})$ can be described as a regular expression whose symbols are linear languages.
Starting from $\mathscr{M}$ we obtain an equivalent grammar $\mathscr{G}$ by the triple construction (cf. [10]). The grammar $\mathscr{G}$ has the following productions: $S \to [q_0, Z_0, q]$ for each $q \in Q$, and

$$[q, A, q_{m+1}] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_m, B_m, q_{m+1}]$$

for each $q, q_1, \ldots, q_{m+1} \in Q$, each $a \in \Sigma \cup \{\varepsilon\}$, and $A, B_1, \ldots, B_m \in \Gamma$, such that $\delta(q, a, A)$ contains $(q_1, B_1 B_2 \ldots B_m)$.
Remember that the initial stack symbol Z_0 appears at the bottom of the pushdown store only. So, we can identify in $\mathscr{M}$ two types of transition rules

$$(q_1, B_1 B_2 \ldots B_m) \in \delta(q, a, A).$$

If $A \neq Z_0$, then either $m = 0$ or $B_i \neq Z_0$, for all $1 \leqslant i \leqslant m$. If $A = Z_0$, then either $m = 0$ or $B_m = Z_0$ and $B_i \neq Z_0$, for all $1 \leqslant i \leqslant m - 1$.

The first type of rules in $\mathscr{M}$ are non-Z_0-rules that result in non-Z_0-productions in $\mathscr{G}$ of the form

$$[q, A, q_{m+1}] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_m, B_m, q_{m+1}]$$

with $A \neq Z_0$ and $B_i \neq Z_0$, for all $1 \leqslant i \leqslant m$, or $[q, A, q_{m+1}] \to a$.

The second type of Z_0-rules in $\mathscr{M}$ result in Z_0-productions in $\mathscr{G}$ of the form

$$[q, Z_0, q_{m+1}] \to a[q_1, B_1, q_2][q_2, B_2, q_3] \cdots [q_m, Z_0, q_{m+1}]$$

with $B_i \neq Z_0$, for all $1 \leqslant i \leqslant m-1$ or $[q, Z_0, q_{m+1}] \to a$.

We now encapsulate the right-hand sides of the production by building super-symbols from symbols. Non-Z_0-productions are changed to

$$[q, A, q_{m+1}] \to [a, [q_1, B_1, q_2], [q_2, B_2, q_3], \ldots, [q_m, B_m, q_{m+1}]]$$

and $[q, A, q_{m+1}] \to [a]$, respectively. Z_0-productions are changed to

$$[q, Z_0, q_{m+1}] \to [a, [q_1, B_1, q_2], \ldots, [q_{m-1}, B_{m-1}, q_m]][q_m, Z_0, q_{m+1}]$$

and $[q, Z_0, q_{m+1}] \to [a]$, respectively. Then the modified grammar $\overline{\mathscr{G}}$ has the following form. Each right-hand side consists either of one Z_0-triple, or one super-symbol or one super-symbol followed by one Z_0-triple.

Since each new turn can only start when the pushdown store is empty up to Z_0, we observe that each super-symbol generates a linear language. We now consider the modified grammar $\overline{\mathscr{G}}$ as a grammar $\mathscr{G}'$ over a finite terminal alphabet consisting of all possible super-symbols. The non-terminals are S and the Z_0-triples. We observe that $\mathscr{G}'$ is right-linear. From $\mathscr{G}'$ we obtain an equivalent regular expression $\mathscr{E}$ by standard techniques. The symbols of $\mathscr{E}$ are the terminal symbols of $\mathscr{G}'$ which in turn are the super-symbols of $\overline{\mathscr{G}}$. Since each super-symbol of $\overline{\mathscr{G}}$ generates a linear language, the language generated by $\mathscr{G}$ can be represented as a regular expression whose symbols describe linear languages. Thus, $N(\mathscr{M}) \in \Gamma_{\mathrm{REG}}(\mathrm{LIN})$. □

The next step is to consider tPDAs that both have to empty their pushdown store and have to return to the initial state in order to start a new turn. Intuitively, in this case no information can be transferred from one turn to the next turn. In fact, the computational power of such devices is not stronger than the computational power of the corresponding one-turn devices.

Lemma 6. *Let $k \in \mathbb{N}$ be a positive integer. A language L is accepted by a (k, q_0, Z_0)-tPDA if and only if L is linear.*

Proof. It is easily observed that any linear language can be accepted by some (k, q_0, Z_0)-tPDA that never reenters the initial state.

Conversely, let $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset\rangle$ be a (k, q_0, Z_0)-tPDA that accepts by empty pushdown store. We define two languages

$$L_1 = \{w | (q_0, w, Z_0) \vdash^* (q_0, \varepsilon, Z_0) \text{ and } \mathscr{M} \text{ performs at most one turn}\},$$
$$L_2 = \{w | (q_0, w, Z_0) \vdash^* (q_0, \varepsilon, \varepsilon) \text{ and } \mathscr{M} \text{ performs at most one turn}\}$$

and observe that both languages are linear. Moreover, $N(\mathscr{M})$ is represented by $L_1^* L_2$.

Let there be words $w \in L_1$ and $v \in L_2$ such that $\mathscr{M}$ accepts wv. Assume that $\mathscr{M}$ performs one turn while processing w. Then the input $w^{k+1}v$ is accepted by $\mathscr{M}$ with at least $k+1$ turns, a contradiction. We conclude that L_1 is a regular language. Since LIN is closed under left concatenation with regular languages, $N(\mathscr{M}) = L_1^* L_2$ is a linear language. □

The reason for the poor computational power of (k, q_0, Z_0)-tPDA is that the devices cannot remember how often subwords from L_1 have been processed. This implies that, whenever at least one turn is performed, an arbitrary number of turns exceeding any k can be performed. But obviously, the mode of acceptance requires that any accepting computation obeys the bound k. Let us call this the *strong acceptance mode*.

Then we define the *weak acceptance mode* as follows: The language accepted by some k-tPDA is $\{w \in \Sigma^* \mid \mathcal{M}$ accepts w with at most k turns$\}$. This means that there may be words accepted by $\mathcal{M}$ with more than k turns, but by definition these words do not belong to the accepted language.

Whether we use the strong or weak acceptance mode makes a difference only for bounded turns if no information is transferred from one turn to the next one. So, we consider (k, q_0, Z_0)-tPDAs under this aspect and obtain immediately a result which is proved similarly to Lemma 6.

Let L be some language and $k \in \mathbb{N}$ be a positive integer, then we define $L^{\leqslant k-1}$ to be the union $L^0 \cup L^1 \cup \cdots \cup L^{k-1}$.

Lemma 7. *Let $k \in \mathbb{N}$ be a positive integer. A language L is accepted by a* (k, q_0, Z_0)-tPDA *with weak acceptance mode if and only if L belongs to $L_1^{\leqslant k-1} L_2$, where L_1, L_2 are linear languages.*

For an unbounded number of turns weak and strong acceptance modes make no difference.

Lemma 8. *A language L is accepted by a* (∞, q_0, Z_0)-tPDA *if and only if L either belongs to the Kleene star closure of the linear languages $\Gamma_*(\mathrm{LIN})$ or to $L_1^* L_2$, where L_1, L_2 are linear languages.*

Proof. Given a language of the form $L_1^* L_2$, where L_1, L_2 are linear languages, it is easy to construct an equivalent (∞, q_0, Z_0)-tPDA. Setting L_1 or L_2 to be the language $\{\varepsilon\}$ shows that any language from $\{L^* \mid L \in \mathrm{LIN}\} \cup \mathrm{LIN} = \Gamma_*(\mathrm{LIN})$ is accepted by a (∞, q_0, Z_0)-tPDA.

Conversely, let $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset\rangle$ be a (∞, q_0, Z_0)-tPDA that accepts by empty pushdown store. Again, we use the two languages L_1 and L_2 defined in the proof of Lemma 6 and observe that both languages are linear. We consider four cases: if L_1 is empty or equals $\{\varepsilon\}$, then $N(\mathcal{M})$ is the linear language $L_2 \in \Gamma_*(\mathrm{LIN})$. If L_2 is empty, then $N(\mathcal{M})$ is empty and, thus, belongs to $\Gamma_*(\mathrm{LIN})$. If L_2 equals $\{\varepsilon\}$, then $N(\mathcal{M}) = L_1^* \in \Gamma_*(\mathrm{LIN})$. Otherwise, $N(\mathcal{M})$ can be represented as $L_1^* L_2$. □

Since we are interested in bridging descriptions of language families by automata and by closures under algebraic operations, we wish to characterize the closure $\Gamma_*(\mathrm{LIN})$. Therefore, we slightly have to weaken the computational power of (∞, q_0, Z_0)-tPDA. To this end, we introduce an additional restriction C, that is, a (∞, q_0, Z_0)-tPDA meets restriction C if and only if, either the only transition that removes the bottom-of-pushdown symbol from the pushdown is $(q_0, \varepsilon, Z_0) \vdash (q, \varepsilon, \varepsilon)$, for some $q \in Q$, or the initial state q_0 is never reached again after the first non-epsilon move.

Now the characterization reads as follows.

Theorem 9. *A language L is accepted by a C-restricted* (∞, q_0, Z_0)-tPDA *if and only if L belongs to the Kleene star closure of the linear languages $\Gamma_*(\mathrm{LIN})$.*

Proof. Let $L \in \Gamma_*(\mathrm{LIN})$, then either $L \in \mathrm{LIN}$ or $L = L_1^*$ with $L_1 \in \mathrm{LIN}$. For any linear language a C-restricted (∞, q_0, Z_0)-tPDA (of the second type) that accepts it can easily be constructed. If $L = L_1^*$ for a linear language L_1,

Table 1
Characterizations of different turn pushdown automata

Restriction	Characterization
k	ULTRALIN
∞	CFL
k, q_0	ULTRALIN
∞, q_0	CFL
k, Z_0	$\Gamma_\cup(\Gamma_\bullet(\mathrm{LIN})) = \mathrm{METALIN}$
∞, Z_0	$\Gamma_{\mathrm{REG}}(\mathrm{LIN})$
k, q_0, Z_0 strong	LIN
k, q_0, Z_0 weak	$L^{\leqslant k-1} L'$ with $L, L' \in \mathrm{LIN}$
∞, q_0, Z_0	$L^* L' + L^*$ with $L, L' \in \mathrm{LIN}$
∞, q_0, Z_0, C	$\Gamma_*(\mathrm{LIN})$

Characterizations of k-turn devices are with respect to the union over all k.

then $L \in \mathscr{L}(C\text{-restricted } (\infty, q_0, Z_0)\text{-tPDA})$, since the language class $\mathscr{L}(C\text{-restricted } (\infty, q_0, Z_0)\text{-tPDA})$ is closed under Kleene star (cf. Lemma 10).

Conversely, let $\mathscr{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ be a C-restricted (∞, q_0, Z_0)-tPDA. If the only transition that removes the bottom-of-pushdown symbol from the pushdown is $(q_0, \varepsilon, Z_0) \vdash (q, \varepsilon, \varepsilon)$, then due to the C-restriction the two languages defined in the proof of Lemma 8 are equal. So, $L_1^* L_2$ belongs to $\Gamma_*(\mathrm{LIN})$. Therefore, $N(\mathscr{M}) \in \Gamma_*(\mathrm{LIN})$. If the initial state q_0 is never reached again after the first non-epsilon move, then the automaton may never start a second turn. So, $N(\mathscr{M}) \in \mathrm{LIN} \subseteq \Gamma_*(\mathrm{LIN})$. □

Table 1 summarizes the characterization results.

4. Closure properties

In this section we consider closure properties of turn bounded PDA languages. By characterization results obtained in the previous section the properties of some families are known. For other families the properties are also of interest in their own. For example, it is natural to ask for the closure properties of closures. The results obtained here are used in the next section in order to compare the computational power of the devices in question. It will turn out that with respect to set inclusion they form a double-stranded hierarchy.

We start with closure properties that already have been used in proofs of the previous section. Therefore, we must not utilize the characterizations.

Lemma 10. *The language family* $\mathscr{L}((\infty, Z_0)\text{-tPDA})$ *is closed under intersection with regular sets*, *union*, *concatenation*, *and Kleene star*.

The families $\mathscr{L}((\infty, q_0, Z_0)\text{-tPDA})$ *and* $\mathscr{L}(C\text{-restricted } (\infty, q_0, Z_0)\text{-tPDA})$ *are closed under Kleene star*.

Proof. First we consider the family $\mathscr{L}((\infty, Z_0)\text{-tPDA})$. Closure under intersection with regular sets follows immediately by the standard Cartesian product construction and by the observation that the resulting automaton is still a (∞, Z_0)-tPDA.

Let $\mathscr{M}_1 = \langle Q_1, \Sigma, \Gamma_1, \delta_1, q_{0,1}, Z_0, F_1\rangle$ and $\mathscr{M}_2 = \langle Q_2, \Sigma, \Gamma_2, \delta_2, q_{0,2}, Z_0, F_2\rangle$ be two (∞, Z_0)-tPDA such that $Q_1 \cap Q_2 = \emptyset$ and $\Gamma_1 \cap \Gamma_2 = \{Z_0\}$. It is easy to construct a (∞, Z_0)-tPDA $\mathscr{M}$ which has a new initial state q_0 and guesses in its first step whether to accept a word from $N(\mathscr{M}_1)$ or from $N(\mathscr{M}_2)$. This proves the closure under union.

To show the closure under concatenation we may assume that $\mathscr{M}_1$ and $\mathscr{M}_2$ accept by empty pushdown store. Define $\mathscr{M} = \langle Q_1 \cup Q_2, \Sigma, \Gamma_1 \cup \Gamma_2, \delta, q_{0,1}, Z_0, \emptyset\rangle$. The transition function δ applies rules from $\mathscr{M}_1$ if the current state is in Q_1 and from $\mathscr{M}_2$ otherwise. Furthermore, we have to ensure that the second automaton can start whenever the first automaton has accepted a subword of the input. To this end, we add the rule $(q_{0,2}, Z_0) \in \delta(q, a, Z_0)$, if $(q', \varepsilon) \in \delta_1(q, a, Z_0)$ with $q, q' \in Q_1$ and $a \in \Sigma \cup \{\varepsilon\}$. Obviously, $\mathscr{M}$ is again a (∞, Z_0)-tPDA.

The construction for the Kleene star closure is similar. We may again assume that $\mathscr{M}_1$ accepts by empty pushdown store. Let $\mathscr{M} = \langle Q_1, \Sigma, \Gamma_1, \delta, q_{0,1}, Z_0, \emptyset\rangle$ and the transition rules of $\mathscr{M}$ be those of $\mathscr{M}_1$ with the following exception. A rule $(q', \varepsilon) \in \delta_1(q, a, Z_0)$ with $q, q' \in Q_1$ and $a \in \Sigma \cup \{\varepsilon\}$ in $\mathscr{M}_1$ is changed to a rule $(q_{0,1}, Z_0) \in \delta(q, a, Z_0)$ in $\mathscr{M}$. Furthermore, the rule $(q_{0,1}, \varepsilon) \in \delta(q_{0,1}, \varepsilon, Z_0)$ is added. It can be observed that $\mathscr{M}$ accepts $N(\mathscr{M}_1)^*$. Moreover, the same construction shows also closure under Kleene star for the language families $\mathscr{L}((\infty, q_0, Z_0)\text{-tPDA})$ and $\mathscr{L}(C\text{-restricted } (\infty, q_0, Z_0)\text{-tPDA})$. □

In the sequel we often will show closure properties of turn bounded automata languages by means of their characterizations. For convenience, we define

$$\mathscr{L}_{\leqslant} = \bigcup_{k=1}^{\infty} \{L^{\leqslant k-1} L' \mid L, L' \in \mathrm{LIN}\}.$$

Obviously, $\mathscr{L}_{\leqslant}$ is the characterization of the family of languages L such that there exists a constant $k \in \mathbb{N}$ and a (k, q_0, Z_0)-tPDA that accepts L with weak acceptance mode (cf. Lemma 7). Similarly, let $\mathscr{L}_* = \{L^* L' \mid L, L' \in \mathrm{LIN}\}$ be the characterization of the family of languages accepted by some (∞, q_0, Z_0)-tPDA (cf. Lemma 8).

The next lemma provides witness languages.

Let $L_{a,b} = \{a^n b^n \mid n \geqslant 1\}$ and $L_{c,d} = \{c^m d^m \mid m \geqslant 1\}$. Further let $L_k = L_{a,b} L_{c,d} \cup L_{c,d}$ and $L_c = L_k\{a^n b^n | n \geqslant 1\}$ and $L_s = (L_{a,b}^* L_{c,d})^*$.

Lemma 11. (1) $L_k \in \mathscr{L}_{\leqslant}$; (2) $L_k \notin \mathrm{LIN}$; (3) $L_k \notin \Gamma_*(\mathrm{LIN})$; (4) $L_k \notin \mathscr{L}_*$; (5) $L_c \notin \mathscr{L}_{\leqslant}$; and (6) $L_s \notin \Gamma_*(\mathrm{LIN})$.

Proof. It is easy to construct a $(2, q_0, Z_0)$-tPDA that accepts L_k with weak acceptance mode. Therefore, $L_k \in \mathscr{L}_{\leqslant}$ follows.

Claim 2 follows immediately by the pumping lemma for linear languages.

Assume now $L_k \in \Gamma_*(\mathrm{LIN})$. Since due to Claim 2 language L_k is not linear, it has a representation $L_k = L^*$ where $L \in \mathrm{LIN}$. A word $z = a^n b^n c^m d^m \in L_k$ has a factorization $z = uv$ with $u \in L$ and $v \in L^+$. Thus, u is of the form a^l, $a^n b^l$, $a^n b^n c^l$, or $a^n b^n c^m d^l$ with $l \geqslant 1$. Since $u \in L$, we know that $z' = uvu \in L_k$ which leads to a contradiction, since each of the above cases implies that z' has a wrong form and, thus, $z' \notin L_k$. This proves Claim 3.

To prove Claim 4 we contrarily assume $L_k = L^* L'$ such that $L, L' \in \mathrm{LIN}$ and $L, L' \neq \{\varepsilon\}$. As above $z = a^n b^n c^m d^m \in L_k$ has a factorization $z = uvw$ with $u \in L$, $v \in L^*$, and $w \in L'$. Furthermore, $z' = uvuw \in L_k$. Again, u is of the form a^l, $a^n b^l$, $a^n b^n c^l$, or $a^n b^n c^m d^l$ with $l \geqslant 1$. If $u = a^l$ and $v \in \{a\}^*$, then z' has more a's than b's which is a contradiction. The remaining cases lead to a wrong form of z' and, thus, also to a contradiction.

Now, let $L_c \in \mathscr{L}_{\leqslant}$ with $L_c = \bigcup_{i=0}^{k-1} L^i L'$, $L, L' \in \mathrm{LIN}$, and a fixed number k. Obviously, $L_c \notin \mathrm{LIN}$ and, thus, $k \geqslant 2$. Then every $z = a^n b^n c^m d^m a^l b^l \in L_c$ has a factorization $z = u_1 u_2 \dots u_{k-1} w$ such that $u_1, u_2 \dots u_{k-1} \in L$ and $w \in L'$. Without loss of generality we may assume that there are at least two strings u_i, u_j such that $u_i \neq \varepsilon$ and $u_j \neq \varepsilon$. Let $\pi : \{1, 2, \dots, k-1\} \to \{1, 2, \dots, k-1\}$ be a permutation. We observe that $z' = u_{\pi(1)} u_{\pi(2)} \dots u_{\pi(k-1)} w \in L_c$. We now distinguish two cases. First, we assume that there is a string $z \in L_c$ such that z has a factorization with $u_1 u_2 \dots u_{k-1} \in a^+ b^+ c^* d^* a^* b^*$. Then there is a permutation π such that z' has a wrong form which leads to a contradiction. On the other hand, if all factorizations of strings $z \in L_c$ imply $u_1 u_2 \dots u_{k-1} \in \{a\}^+$, we can conclude that $L_c = \bigcup_{i=0}^{k-1} L^i L'$ such that $L \subseteq \{a\}^*$. Hence, L is a regular language which implies that $L_c \in \mathrm{LIN}$, since LIN is closed under left concatenation with regular sets. This is a contradiction.

In order to prove Claim 6, we observe $L_s \notin \mathrm{LIN}$. Next, assume contrarily $L_s = L^*$ with $L \in \mathrm{LIN}$. We conclude that all words $z \in L_s$ of the form $z = a^n b^n c^m d^m a^l b^l c^k d^k$ have a factorization $z = uv$ with $u \in L$, $v \in L^+$ and either $u \in a^n b^n$ or $u \in a^n b^n c^m d^m$, since all other factorizations lead to a wrong form due to the fact that $z' = uvu \in L_s$. In the first case we obtain again a contradiction, since the resulting string z' does not end with $c^+ d^+$. In the second case we obtain that L contains all strings of the form $a^n b^n c^m d^m$. Since L is linear and the linear languages are closed under intersection with regular languages, we conclude that $\{a^n b^n c^m d^m \mid n, m \geqslant 1\}$ is a linear language, which is a contradiction. □

Corollary 12. *The language families $\mathscr{L}_{\leqslant}$, $\mathscr{L}_*$, and $\Gamma_*(\mathrm{LIN})$ are not closed under concatenation.*

Proof. Since $L_k \in \mathscr{L}_{\leqslant}$ and $L_{a,b} \in \mathrm{LIN} \subset \mathscr{L}_{\leqslant}$, the assumption that $\mathscr{L}_{\leqslant}$ is closed under concatenation leads to a contradiction, because $L_c \notin \mathscr{L}_{\leqslant}$.

The languages $L_{a,b}$ and $L_{c,d}$ are linear languages and thus in $\Gamma_*(\mathrm{LIN}) \cap \mathscr{L}_*$. Since $L_k \notin \Gamma_*(\mathrm{LIN}) \cup \mathscr{L}_*$, both language families are not closed under concatenation. □

Lemma 13. *The language families $\mathscr{L}_{\leqslant}$, $\mathscr{L}_*$, and $\Gamma_*(\mathrm{LIN})$ are not closed under intersection with regular sets. The family $\mathscr{L}_{\leqslant}$ is not closed under Kleene star.*

Proof. In [5] it has been shown that for any linear, non-regular language L the marked Kleene star $(Lc)^*$ is not even ultralinear.

Since $Lc \in \mathrm{LIN}$ we derive $Lc \in \mathscr{L}_{\leqslant}$. Since $(Lc)^* \notin \mathrm{ULTRALIN}$ we derive $(Lc)^* \notin \mathscr{L}_{\leqslant}$. So, the non-closure under Kleene star follows.

Next, we observe that $L_{a,b} \cup L_{c,d} \in \mathrm{LIN}$. The families $\mathscr{L}_*$ and $\Gamma_*(\mathrm{LIN})$ are both closed under Kleene star. So, the assumption that one of the families is closed under intersection with regular languages implies that $L_k = (L_{a,b} \cup L_{c,d})^* \cap (a^* b^* c^+ d^+)$ belongs to $\mathscr{L}_*$ or $\Gamma_*(\mathrm{LIN})$, respectively. This is a contradiction to Lemma 11.

Clearly, $(L_{a,b} \cup L_{c,d})^3$ belongs to $\mathscr{L}_{\leqslant}$. The assumption that $\mathscr{L}_{\leqslant}$ is closed under intersection with regular languages implies that $L_c = (L_{a,b} \cup L_{c,d})^3 \cap (a^* b^* c^+ d^+ a^+ b^+)$ belongs to $\mathscr{L}_{\leqslant}$ which is a contradiction to Lemma 11. □

Lemma 14. *The language families $\mathscr{L}_{\leqslant}$, $\mathscr{L}_*$, and $\Gamma_*(\mathrm{LIN})$ are not closed under union.*

Proof. We consider the language $L_0 = L_{a,b}^* \cup L_{c,d}^*$ and assume contrarily that $\mathscr{L}_*$ and $\Gamma_*(\mathrm{LIN})$ are closed under union. Then we conclude $L_0 \in \mathscr{L}_*$ and $L_0 \in \Gamma_*(\mathrm{LIN})$.

If $L_0 \in \Gamma_*(\mathrm{LIN})$, then it has a representation $L_0 = L^*$ with $L \in \mathrm{LIN}$, since L_0 is not linear. We consider two words $z_1 = a^n b^n a^n b^n \in L_0$ and $z_2 = c^m d^m c^m d^m \in L_0$. Both words have a factorization $z_1 = u_1 v_1$ and $z_2 = u_2 v_2$ with $u_1, u_2 \in L$ and $v_1, v_2 \in L^+$. We may assume that $u_1 \in a^+ b^+$ and $u_2 \in c^+ d^+$, since other factorizations lead to a wrong form due to the fact that $u_1 v_1 u_1 \in L_0$ and $u_2 v_2 u_2 \in L_0$. Then we conclude that $z' = u_1 v_1 u_2 \in L_0$ which is a contradiction, since z' contains a's and c's.

If $L_0 \in \mathscr{L}_*$, we may assume that $L_0 = L^* L'$ with $L, L' \in \mathrm{LIN}$. By similar considerations concerning z_1 and z_2 we obtain $z_1 = u_1 v_1 w_1$ and $z_2 = u_2 v_2 w_2$ with $u_1, u_2 \in L$, $v_1, v_2 \in L^*$, and $w_1, w_2 \in L'$. Furthermore, $u_1 v_1 u_1 w_1 \in L_0$ and $u_2 v_2 u_2 w_2 \in L_0$. We may assume that $u_1 \in a^+ b^+$ and $u_2 \in c^+ d^+$, since other factorizations lead to a wrong form or to a different number of a's and b's. Again, we can conclude that $u_1 v_1 u_2 w_1 \in L_0$ which is a contradiction.

Finally, we consider the language $L_k \cup L_k'$ with $L_k' = L_{a',b'} L_{c',d'} \cup L_{c',d'}$. We now contrarily assume that $\mathscr{L}_{\leqslant}$ is closed under union. Then we obtain $L_k \cup L_k' \in \mathscr{L}_{\leqslant}$ and $L_k \cup L_k' = \bigcup_{i=0}^{k-1} L^i L'$ with $L, L' \in \mathrm{LIN}$ and a fixed constant $k \in \mathbb{N}$. We consider $z_1 = a^n b^n c^n d^n \in L_k \cup L_k'$ and $z_2 = (a')^m (b')^m (c')^m (d')^m \in L_k \cup L_k'$. Both words have factorizations $z_1 = u_1 \ldots u_i w$ and $z_2 = u_1' \ldots u_j' w'$ with $u_1, \ldots, u_i, u_1', \ldots, u_j' \in L, 1 \leqslant i, j \leqslant k-1$, and $w, w' \in L'$. We may assume that $u_1 \in a^+ b^* c^* d^*$ and $u_1' \in (a')^+ (b')^* (c')^* (d')^*$. Since $u_1, u_1' \in L$, we obtain that $u_1 u_1' w \in L_k \cup L_k'$ which is a contradiction. Thus, $\mathscr{L}_{\leqslant}$ is not closed under union. □

Next, we consider the closure under homomorphisms. Since homomorphisms commute with regular operations, the closures follow more or less immediately. We recall briefly the relevant properties of homomorphisms.

Let $A, B \subseteq \Sigma^*$ and h be a homomorphism. Then, $h(A \cup B) = h(A) \cup h(B)$, $h(AB) = h(A)h(B)$, and $h(A^*) = h(A)^*$.

Corollary 15. *The language families $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$, $\mathscr{L}_{\leqslant}$, $\mathscr{L}_*$, and $\Gamma_*(\mathrm{LIN})$ are closed under homomorphism.*

Proof. Since every language from the above language families admits a representation as a regular expression with linear context-free atoms and LIN is closed under homomorphism, we obtain the claim by the above summarized properties. □

Lemma 16. *The language family $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$ is closed under inverse homomorphism.*

Proof. Let $\mathscr{M}$ be a (∞, Z_0)-tPDA, h be a homomorphism, and $\mathscr{M}'$ be the PDA accepting $h^{-1}(T(\mathscr{M}))$ using the construction given in [10]. It can be observed that this construction does not affect the behavior of the stack. Thus, $\mathscr{M}'$ is also a (∞, Z_0)-tPDA. □

Lemma 17. *The language families $\mathscr{L}_{\leqslant}$, $\mathscr{L}_*$, and $\Gamma_*(\mathrm{LIN})$ are not closed under inverse homomorphism.*

Proof. We consider the linear, but non-regular language $L_0 = \{a^n b^n | n \geqslant 1\} \cup \{b^n a^n | n \geqslant 1\}$. Clearly, $L_0^* L_0$ belongs to $\Gamma_*(\mathrm{LIN})$ and $\mathscr{L}_*$, and, for any $k \geqslant 1$, $L_0^{\leqslant k-1} L_0$ belongs to $\mathscr{L}_{\leqslant}$.

Let $h : \{a, b, c\}^* \to \{a, b\}^*$ be the homomorphism defined by $h(a) = a$, $h(b) = b$, and $h(c) = bb$. We use $h^{-1}(L_0^* L_0)$ and $h^{-1}(L_0^{\leqslant k-1} L_0)$ as witness languages.

In contrast to the assertion we assume that the witness languages are accepted by some tPDA $\mathscr{M}$ characterizing the requested language family, respectively. On infinitely many input words of the form $a^m b^{m-1} c b^{n-1} a^n$, $m, n \geqslant 1$, the automaton $\mathscr{M}$ has to perform at least two turns. Otherwise one could easily construct a classical one-turn PDA accepting the language $\{a^n b^{n+m} a^m \mid m, n \geqslant 1\} = h^{-1}(L_0^* L_0) \cap a^* b^* a^* = h^{-1}(L_0^{\leqslant k-1} L_0) \cap a^* b^* a^*$.

So, the witness languages have the representations $L^* L'$ and $L^{\leqslant k-1} L'$, respectively, where L and L' are linear languages. Thus, a word $z = a^m b^{m-1} c b^{n-1} a^n$ has a factorization $z = uv$, where $|u| \geqslant 1$, $|v| \geqslant 1$, $v \in L'$ and $u \in L^*$ and $u \in L^{\leqslant k-1}$, respectively. This implies that v belongs to the witness languages, too, i.e., $v \in h^{-1}(L_0^* L_0)$ and $v \in h^{-1}(L_0^{\leqslant k-1} L_0)$, respectively. But this is a contradiction since there is no proper suffix v of $a^m b^{m-1} c b^{n-1} a^n$ such that $h(v)$ belongs to $L_0^* L_0$ or to $L_0^{\leqslant k-1} L_0$. □

Table 2 summarizes the closure properties.

Table 2
Closure properties of turn pushdown automata languages

Restriction	Characterization	$\cup$	$\bullet$	$*$	h	h^{-1}	$\cap R$
k	ULTRALIN	+	+	−	+	+	+
∞	CFL	+	+	+	+	+	+
k, q_0	ULTRALIN	+	+	−	+	+	+
∞, q_0	CFL	+	+	+	+	+	+
k, Z_0	METALIN	+	+	−	+	+	+
∞, Z_0	$\Gamma_{\mathrm{REG}}(\mathrm{LIN})$	+	+	+	+	+	+
k, q_0, Z_0 strong	LIN	+	−	−	+	+	+
k, q_0, Z_0 weak	$L^{\leqslant k-1}L'$ with $L, L' \in \mathrm{LIN}$ ($\mathscr{L}_{\leqslant}$)	−	−	−	+	−	−
∞, q_0, Z_0	$L^*L' + L^*$ with $L, L' \in \mathrm{LIN}$ ($\mathscr{L}_*$)	−	−	+	+	−	−
∞, q_0, Z_0, (C)	$\Gamma_*(\mathrm{LIN})$	−	−	+	+	−	−

Characterizations of k-turn devices are with respect to the union over all k.

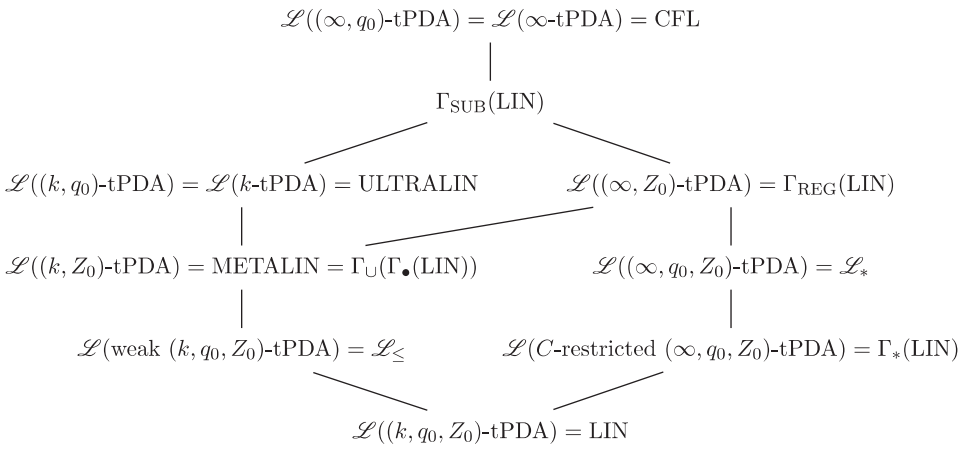

Fig. 1. Inclusion structure. The lines indicate strict inclusions from left to right. All families not linked by a path are pairwise incomparable.

5. Relations between the classes

In this section we present some relations between the considered automata classes with respect to set inclusion. The results reveal a double-stranded hierarchy.

Theorem 18. *All inclusions shown in Fig. 1 are strict. Moreover, language families that are not linked by a path are pairwise incomparable.*

Proof. The strict inclusions LIN $\subset$ METALIN $\subset$ CFL are well known. A proof for the known strict inclusion METALIN $\subset$ ULTRALIN can be found, e.g., in [1].

In order to complete the first strand, LIN $\subset \mathscr{L}_{\leqslant} \subset$ METALIN has to be shown. Since $L_k \in \mathscr{L}_{\leqslant} -$ LIN (Lemma 11) and METALIN is closed under concatenation, but $\mathscr{L}_{\leqslant}$ is not, the strictness of the inclusions follows.

The inclusions on the second strand are easy to see: LIN $\subseteq \Gamma_*(\mathrm{LIN}) \subseteq \mathscr{L}_* \subseteq \Gamma_{\mathrm{REG}}(\mathrm{LIN})$. Their strictness follows from the following facts. First, $\Gamma_*(\mathrm{LIN})$ is closed under Kleene star but LIN is not. Due to Lemma 11 and the closure of $\mathscr{L}_*$ under Kleene star we know that $L_s \in \mathscr{L}_*$ but $L_s \notin \Gamma_*(\mathrm{LIN})$. Furthermore, $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$ is closed under concatenation but $\mathscr{L}_*$ is not.

The inclusion METALIN $\subset \Gamma_{\mathrm{REG}}(\mathrm{LIN})$ is strict because $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$ is closed under Kleene star but METALIN is not.

We next show the incomparableness results by considering three languages. First, $L_k \in \mathscr{L}_{\leqslant}$ but $L_k \notin \mathscr{L}_*$ due to Lemma 11. Second, the language $(Lc)^*$, where $Lc \in$ LIN–REG, from [5], has been shown not to belong to ULTRALIN.

Clearly, $(Lc)^*$ belongs to $\Gamma_*(\mathrm{LIN})$. It is shown in [13] that there exists an ultralinear language $\tilde{L}$ that does not belong to $\mathscr{L}((\infty, Z_0)\text{-tPDA}) = \Gamma_{\mathrm{REG}}(\mathrm{LIN})$. This shows all incomparableness claims.

From [2] we obtain the result $\Gamma_{\mathrm{SUB}}(\mathrm{LIN}) \subset \mathrm{CFL}$, where Γ_{SUB} denotes the substitution closure. Clearly, $\Gamma_{\mathrm{REG}}(\mathrm{LIN}) \subseteq \Gamma_{\mathrm{SUB}}(\mathrm{LIN})$ and $\mathrm{ULTRALIN} \subseteq \Gamma_{\mathrm{SUB}}(\mathrm{LIN})$ (cf. [2,6]). The strictness of the inclusions can be shown with the above languages $\tilde{L}$ and $(Lc)^*$. □

We conclude with the observation that for both a bounded and an unbounded number of turns, requiring to empty the pushdown store is a strictly stronger condition than requiring to reenter the initial state (cf. Fig. 1).

6. Parsing

It is known that metalinear languages can be parsed in quadratic time due to the fact that metalinear languages are accepted by deterministic two-way cellular automata [12] whose languages in turn can be recognized with $\mathrm{O}(n^2)$ time and $\mathrm{O}(n)$ space [11,16]. A recognition time of $\mathrm{O}(n^2)$ for $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$ has been stated (without proof) in [17]. For the sake of completeness, we are now going to present an algorithm which parses languages from $\Gamma_{\mathrm{REG}}(\mathrm{LIN})$ in $\mathrm{O}(n^2)$ time. We follow an idea of [9].

A production in a context-free grammar is called *linear*, if its right-hand side contains at most one non-terminal. A non-terminal in a context-free grammar is called linear, if all productions with this non-terminal on the left-hand side are linear. It can be observed that in a linear grammar all non-terminals are linear. A linear grammar is said to be in linear normal form, if all productions are of the form $A \to a$, $A \to aB$, and $A \to Ba$ for $A, B \in N$ and $a \in T$. It can be observed that every ε-free linear language can be generated by some grammar in linear normal form.

Since linear productions have at most one non-terminal on their right-hand side, the following simplification of the Cocke–Younger–Kasami algorithm can be made. Confer also [8, Algorithm 12.4.4].

Input: A context-free grammar $\mathscr{G}$ in linear normal form and an input string $w = a_1 \dots a_n$.

Output: A parse table T for w such that $t_{i,j}$ contains A if and only if $A \Rightarrow^+ a_i a_{i+1} \dots a_{i+j-1}$.

Method: Set $t_{i,1} = \{A \mid A \to a_i\}$ for all $i = 1, \dots, n$. Assume that $t_{i,k}$ has been computed for some $k < n$ and all $i = 1, \dots, n$. Compute

$$t_{i,k+1} = \{A \mid A \to a_i B, B \in t_{i+1,k}\} \cup \{A \mid A \to Ba_{i+k}, B \in t_{i,k}, i + k \leqslant n\}.$$

The correctness can be shown by induction on j. It can be observed that the algorithm has time complexity $\mathrm{O}(n^2)$ and that $S \in t_{1,n}$ if and only if $w \in L(\mathscr{G})$.

A context-free grammar is called *superlinear*, if all linear productions contain only linear non-terminals on the right-hand side, and non-linear productions are of the form $A \to BC$, where C is linear. A language is called superlinear, if it can be generated by some superlinear grammar. A superlinear grammar is said to be in superlinear normal form, if all linear productions are of the form $A \to a$, $A \to aB$, $A \to Ba$, and $A \to B$, for $A, B \in N$ and $a \in T$, and all non-linear productions are of the form $A \to BC$. It can be observed that every ε-free superlinear language can be generated by some superlinear grammar in superlinear normal form.

The special form of non-linear productions in a superlinear grammar implies that every word generated can be represented as concatenation of subwords which themselves are generated by linear grammars. Thus, it is possible to compute the parse tables of the subwords with the above algorithm and then to compute a parse table for the concatenation of these subwords.

Input: A context-free grammar in superlinear normal form and an input string $w = a_1 \cdots a_n$.

Output: A parse table T for w such that $t_{i,j}$ contains A if and only if $A \Rightarrow^+ a_i a_{i+1} \dots a_{i+j-1}$, and a parse table R such that r_j contains A if and only if $A \Rightarrow^+ a_1 \dots a_j$.

Method: Let $N_L \subseteq N$ denote the set of linear non-terminals. Compute T the same way as in the above algorithm, but consider only non-terminals from N_L. Then, for $j = 1, \dots, n$, compute

$$r_j = \{A \mid A \to B, B \in t_{1,j}\} \cup \{A \mid \exists k < j : A \to BC \text{ such that } B \in r_k \text{ and } C \in t_{k+1,j-k}\}.$$

The correctness can be again shown by induction on j. Furthermore, the time complexity is $\mathrm{O}(n^2)$ and $S \in r_n$ if and only if $w \in L(\mathscr{G})$.

Theorem 19. *Every language $L \in \Gamma_{\mathrm{REG}}(\mathrm{LIN})$ is parsable in $\mathrm{O}(n^2)$ time.*

Proof. It is easily observed that every language $L \in \Gamma_{\mathrm{REG}}(\mathrm{LIN})$ can be represented as the substitution τ of linear languages in a regular set R. Thus, let $R \in \mathrm{REG}$ and $\tau(a) = L_a \in \mathrm{LIN}$ for every element a of the alphabet. Then, $L = \tau(R)$. Now, let $\mathcal{G}_R$ be a left-linear grammar generating R and $\mathcal{G}_a$ be linear grammars generating L_a. By substituting every terminal a in $\mathcal{G}_R$ by the start symbol of $\mathcal{G}_a$, we obtain a superlinear grammar $\mathcal{G}$ generating L. Due to the above algorithm we can observe that L can be parsed in quadratic time. □

Corollary 20. *Every language* $L \in \mathscr{L}_{\leqslant}$ ($L \in \mathscr{L}_*$ *or* $L \in \Gamma_*(\mathrm{LIN})$) *is parsable in* $\mathrm{O}(n^2)$ *time.*

References

[1] L. Balke, K.H. Böhling, Einführung in die Automatentheorie und Theorie formaler Sprachen, BI Wissenschaftsverlag, Mannheim, 1993.
[2] J. Berstel, Transductions and Context-Free Languages, Teubner, Stuttgart, 1979.
[3] E. Bertsch, M.J. Nederhof, Regular closure of deterministic languages, SIAM J. Comput. 29 (1999) 81–102.
[4] N. Chomsky, M.P. Schützenberger, The algebraic theory of context-free languages, in: Computer Programming and Formal Systems, North-Holland, Amsterdam, 1963, pp. 118–161.
[5] S. Ginsburg, E.H. Spanier, Finite-turn pushdown automata, SIAM J. Comput. 4 (1966) 429–453.
[6] S. Ginsburg, E.H. Spanier, Derivation-bounded languages, J. Comput. System Sci. 2 (1968) 228–250.
[7] S.A. Greibach, An infinite hierarchy of context-free languages, Journal of Assoc. Comput. Mach. 16 (1969) 91–106.
[8] M.A. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading, Massachusetts, USA, 1978.
[9] C. Herzog, Some CFL's parsable in quadratic time, unpublished manuscript.
[10] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Language, and Computation, Addison-Wesley, Reading, MA, 1979.
[11] O.H. Ibarra, T. Jiang, H. Wang, Parallel parsing on a one-way linear array of finite-state machines, Theoret. Comput. Sci. 85 (1991) 53–74.
[12] M. Kutrib, Automata arrays and context-free languages, in: C. Martín-Vide, V. Mitrana (Eds.), Where Mathematics, Computer Science and Biology Meet, Kluwer Academic Publishers, Dordrecht, 2001, pp. 139–148.
[13] A. Malcher, On recursive and non-recursive trade-offs between finite-turn pushdown automata, in: Descriptional Complexity of Formal Systems (DCFS 2005), Rapporto Tecnico 06-05, Università degli Studi di Milano, 2005, pp. 215–226.
[14] E. Moriya, T. Tada, On the space complexity of turn bounded pushdown automata, Internat. J. Comput. Math. 80 (2003) 295–304.
[15] A. Salomaa, Formal Languages, Academic Press, New York, 1973.
[16] A.R. Smith III, Real-time language recognition by one-dimensional cellular automata, J. Comput. System Sci. 6 (1972) 233–253.
[17] P. Strnad, Turing machine recognition, in: Mathematical Foundations of Computer Science, Slovak Academy of Sciences, Bratislava, 1973, pp. 331–332.
[18] D.A. Workman, Turn-bounded grammars and their relation to ultralinear languages, Inform. and Control 32 (1976) 188–200.