Input-Driven Multi-counter Automata

Martin Kutrib[✉], Andreas Malcher, and Matthias Wendlandt

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{kutrib,andreas.malcher,matthias.wendlandt}@informatik.uni-giessen.de

Abstract. The model of deterministic input-driven multi-counter automata is introduced and studied. On such devices, the input letters uniquely determine the operations on the underlying data structure that is consisting of multiple counters. We study the computational power of the resulting language families and compare them with known language families inside the Chomsky hierarchy. In addition, it is possible to prove a proper counter hierarchy depending on the alphabet size. This means that any input alphabet induces an upper bound which depends on the alphabet size only, such that $k + 1$ counters are more powerful than k counters as long as k is less than this bound. The hierarchy interestingly collapses at the level of the bound. Furthermore, we investigate the closure properties of the language families. Finally, the undecidability of the emptiness problem is derived for input-driven two-counter automata.

1 Introduction

Multi-counter automata are finite state automata equipped with multiple counters which can be incremented, decremented, and tested for zero. It is well known that general one-way deterministic two-counter automata are computationally universal, that is, they can simulate Turing machines [17]. However, the latter simulation may need an unbounded amount of space. Hence, deterministic space-bounded as well as time-bounded multi-counter automata have been considered in [7] where, in particular, the case when the available time is restricted to real-time is studied. The authors establish in this case an infinite and strict counter hierarchy as well as positive and negative closure results. The generalization to multi-counter automata that may work nondeterministically as well as may use two-way motion on the input tape has been done in [8]. Since one-counter automata can be seen as a special case of pushdown automata, multi-counter automata may be considered a special case of multi-pushdown automata introduced in [6].

A recently introduced restriction to pushdown automata which turned out to provide nice closure properties and decidability questions is the requirement to work in an *input-driven* way. This means that input-driven pushdown automata are ordinary pushdown automata where the actions on the pushdown store are dictated by the input symbols. In particular, if an input symbol forces the machine to pop a symbol from the empty pushdown store, the computation

M. Hospodár and G. Jirásková (Eds.): CIAA 2019, LNCS 11601, pp. 197–208, 2019.
https://doi.org/10.1007/978-3-030-23679-3_16

continues with empty pushdown store. This variant of pushdown automata has originally been introduced in 1980 by Mehlhorn [16] and further investigations have been done in 1985 by von Braunmühl and Verbeek [5]. The results of both papers comprise the equivalence of nondeterministic and deterministic models and the proof that the membership problem is solvable in logarithmic space. The model has been revisited under the name of visibly pushdown automata in 2004 [1]. Complexity results on the model are summarized in the survey [18]. An input-driven variant of one-counter automata has been introduced in [2] and two recent papers [9,12] examine algebraic and logical aspects of input-driven counter automata. The above-mentioned generalization to multi-pushdown automata in terms of input-driven devices is described in [15] where several additional restrictions are put on the general model in order to obtain manageable models with positive closure properties and decidable questions. Finally, we mention that the computational power of input-driven automata using the storage medium of a stack and a queue, respectively, have been investigated in [3,13].

In this paper, we will introduce and investigate the model of input-driven multi-counter automata which are basically the input-driven variant of the real-time multi-counter automata discussed in [7]. It should be noted that this model is different from the model of "input-driven pushdown automata with counters" recently introduced by Ibarra [11]. This model is basically an input-driven pushdown automaton with additional reversal-bounded counters (see also [10]), where the input symbols govern the behavior on the pushdown store, but not necessarily on the counters. In contrast, our model has a counter update function which solely depends on the input alphabet. The paper is organized as follows. In the next section we introduce the necessary notations on multi-counter automata and their input-driven versions. In Sect. 3 we study the computational capacity of input-driven multi-counter automata and their relation to the language families of the Chomsky hierarchy. Then, a hierarchy on the number of counters is established that interestingly depends on the size of the input alphabet. This means that every alphabet size n determines a bound $f(n)$ such that k counters with $1 \leq k < f(n)$ are less powerful that $k+1$ counters, but any number of counters larger than $f(n)$ is as powerful as $f(n)$ counters. Sections 4 and 5 are devoted to investigating the closure properties of and decidability questions for input-driven multi-counter automata. The main result in the latter section is that already two input-driven counters are sufficient to obtain that all usually studied decidability questions are undecidable, whereas all but one of the questions is decidable for input-driven one-counter automata.

2 Preliminaries

Let Σ^* denote the set of all words over the finite alphabet Σ. The *empty word* is denoted by λ and the *reversal* of a word w by w^R. For the *length* of w we write $|w|$. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions.*

Let $k \geq 0$ be a natural number. A (one-way) deterministic k-counter automaton (DCA(k)) is a finite automaton having a single read-only input tape. In addition, it is equipped with k counters. At the outset of a computation the counter automaton is in the designated initial state, the counters are set to zero, and the head of the input tape scans the leftmost input symbol. Dependent on the current state, the currently scanned input symbol, and the information whether the counters are zero or not, the counter automaton changes its state, increases or decreases the counters or leaves the counters unchanged, and moves the input head one square to the right. The automata have no extra output tape but the states are partitioned into accepting and rejecting states.

A counter automaton is called *input-driven* if the input symbols currently read define the next action on the counters. To this end, we assume that each input symbol is associated with actions to be applied to the counters. Let Σ be the input alphabet. Then $\alpha\colon \Sigma \to \{-1,0,1\}^k$ gives these actions, where the ith component $\alpha(x)_i$ of $\alpha(x)$, for $1 \leq i \leq k$ and $x \in \Sigma$, is added to the current value of counter i. The subtraction is in natural numbers, that is, decreasing a counter value 0 gives counter value 0. This behavior is in line with the definition of input-driven pushdown automata that may pop from the empty pushdown store leaving the pushdown store empty. For any $x \geq 0$ we define the function $\mathrm{sg}(0) = \perp$ and $\mathrm{sg}(x) = +$ for $x \geq 1$.

An *input-driven counter automaton with* $k \geq 0$ *counters* (IDCA(k)) *is a system* $M = \langle Q, \Sigma, k, q_0, F, \alpha, \delta\rangle$, where Q is the finite set of *internal states*, Σ is the finite set of *input symbols*, $k \geq 0$ is the *number of counters*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, $\alpha\colon \Sigma \to \{-1,0,1\}^k$ is the *counter update function*, and $\delta\colon Q \times \Sigma \times \{+,\perp\}^k \to Q$ is the partial transition function that determines the successor state dependent on the current state, the current input symbol, and the current statuses of the counters (+ indicates a positive value and $\perp$ a zero).

A *configuration* of an IDCA(k) $M = \langle Q, \Sigma, k, q_0, F, \alpha, \delta\rangle$ is a $(k+2)$-tuple $(q, w, c_1, c_2, \ldots, c_k)$, where $q \in Q$ is the current state, $w \in \Sigma^*$ is the unread part of the input, and $c_i \geq 0$ is the current value of counter i, $1 \leq i \leq k$. The *initial configuration* for input w is set to $(q_0, w, 0, 0, \ldots, 0)$. During the course of its computation, M runs through a sequence of configurations. One step from a configuration to its successor configuration is denoted by $\vdash$. Let $q, q' \in Q$, $a \in \Sigma$, $w \in \Sigma^*$, and $c_i \geq 0$, $1 \leq i \leq k$. We set

$$(q, aw, c_1, c_2, \ldots, c_k) \vdash (q', w, c_1 + \alpha(a)_1, c_2 + \alpha(a)_2, \ldots, c_k + \alpha(a)_k)$$

if and only if $\delta(q, a, \mathrm{sg}(c_1), \mathrm{sg}(c_2), \ldots, \mathrm{sg}(c_k)) = q'$ (recall that the subtraction is in natural numbers). As usual, we define the reflexive and transitive closure of $\vdash$ by $\vdash^*$.

The language accepted by the IDCA(k) M is the set $L(M)$ of words for which there exists some computation beginning in the initial configuration and halting in a configuration in which the whole input is read and an accepting state is entered. Formally:

$$L(M) = \{\, w \in \Sigma^* \mid (q_0, w, 0, 0, \ldots, 0) \vdash^* (q, \lambda, c_1, c_2, \ldots, c_k) \text{ with } q \in F, c_i \geq 0 \text{ for } 1 \leq i \leq k \,\}.$$

So, an input-driven k-counter automaton is a realtime device since it cannot perform stationary moves. It halts within n steps on inputs of length n. For each counter C_i, the definitions imply the partition of the input alphabet into the sets $\Sigma_D^{(i)}$, $\Sigma_R^{(i)}$, and $\Sigma_N^{(i)}$ that control the actions increase or drive (D), decrease or reverse (R), and leave unchanged or neutral (N) of counter C_i. Such a partition is called a *signature*.

The family of all languages which can be accepted by some device X is denoted by $\mathscr{L}(\mathsf{X})$.

To clarify our notion we continue with an example.

Example 1. The language $L = \{abb\bar{a}^3\bar{b}\bar{b}a^5bb\bar{a}^7\bar{b}\bar{b}\cdots a^{4n+1}b \mid n \geq 0\}$ is non-semilinear and accepted by the IDCA(2) $M = \langle Q, \Sigma, 2, q_0, F, \alpha, \delta\rangle$ with state set $Q = \{q_0, q_1, q_2, q_3, q_4\}$, final states $F = \{q_2\}$, counter update function defined by $\alpha(a) = (-1, 1)$, $\alpha(b) = (0, 1)$, $\alpha(\bar{a}) = (1, -1)$, $\alpha(\bar{b}) = (1, 0)$, and transition function

1. $\delta(q_0, a, \bot, \bot) = q_1$
2. $\delta(q_1, b, \bot, +) = q_2$
3. $\delta(q_2, b, \bot, +) = q_3$
4. $\delta(q_3, \bar{a}, \bot, +) = q_3$
5. $\delta(q_3, \bar{a}, +, +) = q_3$
6. $\delta(q_3, \bar{b}, +, \bot) = q_4$
7. $\delta(q_4, \bar{b}, +, \bot) = q_1$
8. $\delta(q_1, a, +, \bot) = q_1$
9. $\delta(q_1, a, +, +) = q_1$

The IDCA(2) M uses its second counter to store the number of consecutive a's. The following two b's are used to increment the counter by two in order to match the number of $\bar{a}$'s in the following block. The comparison is made by decreasing the second counter on $\bar{a}$'s. Simultaneously, the first counter is increased to store the number of $\bar{a}$'s for the verification of the next block length. The addition of two is done while reading two symbols $\bar{b}$'s. Similarly the length of an $\bar{a}$ block is compared with the length of the following a block. These comparisons are done alternately.

The correct format of the input is checked in the states.

Finally, the total length of an accepted input is $(2n+2)^2 - 2$ and, thus, the language is not semilinear. ■

3 Computational Capacity

We start the investigation of the computational capacity of input-driven counter automata by considerations on unary languages. Example 1 shows that even two counters are sufficient to push the power of input-driven counter automata beyond the edge of semilinearity and, thus, context-freeness. However, to this end non-unary witness languages have to be used. In fact, for unary languages any number of counters does not help to accept a non-regular language.

Proposition 2. *Any unary language accepted by some IDCA is regular.*

Proof. Let $M = \langle Q, \{a\}, k, q_0, F, \alpha, \delta\rangle$ be an IDCA(k) accepting a unary language $L(M) \subseteq a^*$. If $k = 0$, then M is a finite automaton and the accepted language is regular. If $k > 0$, the signature of any counter C_i, $1 \leq i \leq k$, consists of one singleton and two empty sets. If $\Sigma_R^{(i)}$ or $\Sigma_N^{(i)}$ is non-empty then M will never increase counter C_i from its initial value zero. So, counter C_i is useless and can be omitted. If $\Sigma_D^{(i)}$ is non-empty then counter C_i is never decreased to zero once it has been increased to one. This fact can be remembered in the state such that counter C_i can be omitted in this case either. In this way all counters can be omitted and we end up in a finite automaton that accepts $L(M)$. $\square$

Clearly, any regular language is accepted by some IDCA(0). So, for any $k \geq 0$ the family of unary languages accepted by IDCA(k) coincides with the family of regular languages. Moreover, any IDCA can be simulated by a deterministic linear bounded automaton in a straightforward way. This implies that the family of languages accepted by IDCA is included in the family of deterministic context-sensitive languages. The inclusion is even strict, since the (deterministic) (linear) context-free language $\{ a^n\$a^n \mid n \geq 0 \}$ is not accepted by any IDCA.

Lemma 3. *Language $L = \{ a^n\$a^n \mid n \geq 0 \}$ is not accepted by any IDCA.*

Proof. Contrarily assume that there is some IDCA(k) $M = \langle Q, \Sigma, k, q_0, F, \alpha, \delta \rangle$ that accepts L.

We give evidence that the counters cannot help. Let C_i, $1 \leq i \leq k$, be a counter. If $a \in \Sigma_R^{(i)}$ or $a \in \Sigma_N^{(i)}$ then M will increase counter C_i at most once on reading the \$. This fact can be remembered in the state and counter C_i can be omitted.

If $a \in \Sigma_D^{(i)}$ and $n \geq 2$ then counter C_i is never decreased to zero once two a's have been read from the input. This fact can be remembered in the state as well such that counter C_i can be omitted in this case either.

In this way all counters can be omitted and we end up in a finite automaton that accepts L. This is a contradiction since L is not regular. $\square$

The next corollary summarizes the relationships with the linguistic families of the Chomsky hierarchy.

Corollary 4. *Let $k \geq 1$ be an integer. Then the family of regular languages is strictly included in $\mathscr{L}$(IDCA(k)) which, in turn, is strictly included in the family of deterministic context-sensitive languages.*

For $k \geq 2$, the family $\mathscr{L}$(IDCA(k)) is incomparable with the family of (deterministic) (linear) context-free languages.

Next, we turn to examine the power of the number of counters.

Lemma 5. *Let Σ be an m-symbol alphabet and $k = 3^m - 2^{m+1} + 1$. Then any IDCA($k + i$), $i \geq 1$, can be simulated by an IDCA(k).*

Proof. Let $i \geq 1$ and $M = \langle Q, \Sigma, k + i, q_0, F, \alpha, \delta \rangle$ be an IDCA($k + i$).

The proof of Proposition 2 revealed that a counter whose signature does not associate an increase *and* a decrease operation with some alphabet symbol can safely be omitted. So, any useful signature has non-empty sets Σ_R and Σ_D. There are 3^m different signatures of Σ. From these, 2^m signatures have an empty set Σ_R. Another 2^m signatures have an empty set Σ_D. Moreover, there is exactly one signature with both sets Σ_R and Σ_D empty. Therefore, there are at most $3^m - 2^{m+1} + 1$ different useful signatures.

Clearly, if two counters of M have the same signature, one of them can be omitted. The same is true for a counter with a useless signature. We conclude that at least i counters can be removed from M without affecting the language accepted. □

In particular, Lemma 5 shows that any counter hierarchy for IDCA necessarily collapses at a level that is solely determined by the alphabet size. Next, we turn to show that, in fact, these hierarchies exist.

Theorem 6. *Let $m \geq 2$ be an integer. For $1 < k \leq 3^m - 2^{m+1} + 1$, the family of languages accepted by IDCA$(k-1)$ over an alphabet of size m is strictly included in the family of languages accepted by IDCA(k) over an alphabet of size m.*

Proof. Let $\Sigma = \{a_1, a_2, \dots, a_m\}$ be some alphabet of size $m \geq 2$. The proof of Lemma 5 showed that there are $k_{max} = 3^m - 2^{m+1} + 1$ different useful signatures $S_i = \big(\Sigma_D^{(i)}, \Sigma_R^{(i)}, \Sigma_N^{(i)}\big)$, $1 \leq i \leq k_{max}$. For each signature S_i language $L_i \subseteq \Sigma^*$ is defined as

$$L_i = \{\, w_1 w_2 w_3 \mid w_1 \in \big(\Sigma_D^{(i)}\big)^+, w_2 \in \big(\Sigma_N^{(i)}\big)^*, w_3 \in \big(\Sigma_R^{(i)}\big)^+, |w_3| = |w_1| + 1 \,\}.$$

First, we show that, for any subset $I = \{i_1, i_2, \dots, i_k\} \subseteq \{1, 2, \dots, k_{max}\}$, language $L_I = \bigcup_{i \in I} (L_i)^+$ is accepted by some IDCA(k) M. To this end, the counter update function α of M associates signature S_{i_j} with counter C_j, for $i_j \in I$. In this way, when starting with counter value zero, for any input factor $w_1 w_2 w_3$ from L_{i_j}, counter C_j is incremented to $|w_1|$ while M processes the prefix w_1 of the factor. On w_2 the value of counter C_j is unchanged. Since $|w_3| = |w_1| + 1$, counter C_j is decremented to value zero for the first time when only one input symbol of the factor is left. The next transition of M on counter value zero drives M into an accepting state while the value of counter C_j remains zero. Therefore, M accepts if the input has been processed entirely. Otherwise it repeats the process for the next input factor from L_{i_j}. Moreover, since $\Big(\big(\Sigma_D^{(i_j)}\big)^+ \big(\Sigma_N^{(i_j)}\big)^* \big(\Sigma_R^{(i_j)}\big)^+\Big)^+$ is regular, M can determine in its states whether the input has the format of all words in $(L_{i_j})^+$. Since M has as many counters as languages are joined to L_I, an input can be accepted if and only if it belongs to L_I.

Second, we show that, for any subset $I = \{i_1, i_2, \dots, i_k\} \subseteq \{1, 2, \dots, k_{max}\}$, language L_I is not accepted by any IDCA$(k-1)$. In contrast to the assertion assume that there is some subset $I = \{i_1, i_2, \dots, i_k\}$ and some IDCA$(k-1)$ $M = \langle Q, \Sigma, k-1, q_0, F, \alpha, \delta \rangle$ that accepts L_I. Since L_I is the union of k languages which in turn are defined by k signatures, but M has only $k-1$ counters, there is at least one of the joined languages, say $(L_j)^+$, whose underlying signature $S_j = \big(\Sigma_D^{(j)}, \Sigma_R^{(j)}, \Sigma_N^{(j)}\big)$ does not appear as the signature of any of the counters.

In order to obtain a contradiction, we next turn to construct an input word $\varphi \in (L_j)^+$ that fools all counters of M simultaneously. To this end, let

$$\Sigma_D^{(j)} = \{x_1, x_2, \ldots, x_p\}, \Sigma_N^{(j)} = \{y_1, y_2, \ldots, y_q\}, \text{ and } \Sigma_R^{(j)} = \{z_1, z_2, \ldots, z_r\}.$$

The word φ is the concatenation of r words $u_1, u_2, \ldots, u_r$ from L_j that are as follows. Let $c > r + 2$ be a fixed constant. Then

$$u_i = (x_1 x_2 \cdots x_p)^c z_i^{c \cdot p + 1}, \text{ for } 1 \le i \le r - 1.$$

So, the length of u_i is $2 \cdot c \cdot p + 1$. Now we set $c_r = (r - 1)(2 \cdot c \cdot p + 1) + p + |Q|$ and $s = (p + 1) \cdot c_r$, and define

$$u_r = (x_1 x_2 \cdots x_p)^{c_r} y_1^{2 \cdot s} y_2^{2^2 \cdot s} \cdots y_q^{2^q \cdot s} z_r^{c_r \cdot p + 1}.$$

Next we determine how the counters of M evolve on input $\varphi = u_1 u_2 \cdots u_r$ that belongs to $L(M)$. To this end, we distinguish three cases dependent on the signatures of the counters.

Case 1. Let $\Sigma_N^{(j)} \not\subseteq \Sigma_N^{(i)}$ for some counter C_i.

To determine how counter C_i evolves on input φ we consider the greatest index ℓ from $\{1, 2, \ldots, q\}$ such that $y_\ell \notin \Sigma_N^{(i)}$.

The length of the prefix of φ up to but not including the factor $y_\ell^{2^\ell \cdot s}$ from u_r is $(r-1)(2 \cdot c \cdot p + 1) + c_r \cdot p + 2 \cdot s + 2^2 \cdot s + \cdots + 2^{\ell-1} \cdot s < s + 2 \cdot s + 2^2 \cdot s + \cdots + 2^{\ell-1} \cdot s = (2^\ell - 1) \cdot s$. Therefore, after processing the prefix, the value of counter C_i is at most $(2^\ell - 1) \cdot s$.

If $y_\ell \in \Sigma_R^{(i)}$ then the value of counter C_i is decremented to zero after processing the following factor $y_\ell^{2^\ell \cdot s}$. Furthermore, since ℓ has been chosen to be the greatest index, the value of counter C_i remains zero until the first symbol z_r appears in the input. Dependent on whether z_r belongs to $\Sigma_D^{(i)}$ or $\Sigma_N^{(i)} \cup \Sigma_R^{(i)}$ the value of counter C_i increases on the remaining input suffix $z_r^{c_r \cdot p + 1}$ or remains zero. In both cases the status of counter C_i does not change on the last $c_r \cdot p$ input symbols.

If $y_\ell \in \Sigma_D^{(i)}$ then the value of counter C_i is at least $2^\ell \cdot s$ after processing the following factor $y_\ell^{2^\ell \cdot s}$. Furthermore, since ℓ has been chosen to be the greatest index, the value of counter C_i does not change until the first symbol z_r appears in the input. Moreover, since $c_r \cdot p + 1 < 2^\ell \cdot s$ the status of counter C_i does not change on the last $c_r \cdot p$ input symbols.

Case 2. Let $\Sigma_N^{(j)} \subseteq \Sigma_N^{(i)}$ and $\Sigma_D^{(j)} \not\subseteq \Sigma_D^{(i)}$ for some counter C_i.

The length of the prefix $u_1 u_2 \cdots u_{r-1}$ of φ is $(r - 1)(2 \cdot c \cdot p + 1)$. Therefore, after processing the prefix, the value of counter C_i is at most $(r - 1)(2 \cdot c \cdot p + 1)$. Since there is at least one symbol from $\{x_1, x_2, \ldots, x_p\}$ that does not belong to $\Sigma_D^{(i)}$, the value of counter C_i increases by at most $c_r \cdot (p - 1)$ on processing the following factor $(x_1 x_2 \cdots x_p)^{c_r}$. This gives a counter value of at most $(r - 1)(2 \cdot c \cdot p + 1) + c_r \cdot (p - 1) = c_r - p - |Q| + c_r \cdot p - c_r = c_r \cdot p - p - |Q|$.

Dependent on whether z_r belongs to $\Sigma_D^{(i)} \cup \Sigma_N^{(i)}$ or $\Sigma_R^{(i)}$ the value of counter C_i increases or remains unchanged, or decreases by $c_r \cdot p + 1$ on the remaining input suffix $z_r^{c_r \cdot p+1}$. In the former case, clearly, the status of counter C_i does not change on the last $c_r \cdot p$ input symbols. In the latter case the counter is decreased to zero after processing at most $c_r \cdot p - p - |Q|$ input symbols z_r. So, the status of counter C_i does not change on the last $p + |Q| + 1$ input symbols.

Case 3. Let $\Sigma_N^{(j)} \subseteq \Sigma_N^{(i)}$ and $\Sigma_D^{(j)} \subseteq \Sigma_D^{(i)}$ for some counter C_i.

In this case we know that at least one of the inclusions is strict and obtain $\Sigma_R^{(i)} \subset \Sigma_R^{(j)}$. Let ℓ be the greatest index from $\{1, 2, \ldots, r\}$ such that $z_\ell \notin \Sigma_R^{(i)}$.

Since counter C_i increases on all input factors $x_1 x_2 \cdots x_p$ by p, after processing the prefix $(x_1 x_2 \cdots x_p)^c$ of u_ℓ if $\ell < r$ or $(x_1 x_2 \cdots x_p)^{c_r}$ of u_r if $\ell = r$, the counter value is at least $c \cdot p$ in the former and at least $c_r \cdot p$ in the latter case. Since $z_\ell \notin \Sigma_R^{(i)}$, the value does not decrease on suffix $z_\ell^{c \cdot p+1}$ of u_ℓ if $\ell < r$ or $z_\ell^{c_r \cdot p+1}$ of u_r if $\ell = r$.

If $\ell = r$ this implies immediately that the status of counter C_i does not change on the last $c_r \cdot p$ input symbols.

If $\ell < r$, the value of counter C_i is at least $c \cdot p$ after processing factor u_ℓ of φ. The value increases by $c \cdot p$ and subsequently possibly decreases by $c \cdot p + 1$ on each of the remaining $r - \ell - 1$ factors $u_{\ell+1}, u_{\ell+2}, \ldots, u_{r-\ell-1}$. Finally, on the prefix $(x_1 x_2 \cdots x_p)^{c_r}$ of u_r it increases by $c_r \cdot p$. At that time its value is at least $c \cdot p - (r - \ell - 1) + c_r \cdot p$. Recall that $p, r \geq 1$ since S_j is useful. Since $\Sigma_N^{(j)} \subseteq \Sigma_N^{(i)}$, it decreases by at most $c_r \cdot p + 1$ on the remaining input suffix. By $c \cdot p - (r - \ell - 1) + c_r \cdot p > c \cdot p - r + c_r \cdot p > (r+2) \cdot p - r + c_r \cdot p > 2 + c_r \cdot p$ we conclude that counter C_i is not decremented to zero on the suffix and, thus, the status of counter C_i does not change on the last $c_r \cdot p$ input symbols. This concludes Case 3.

Cases 1 to 3 show that the status of all counters of M at least do not change on the last $p + |Q| \geq 1 + |Q|$ input symbols. Since these input symbols are all the same, that is, they are z_r, automaton M enters some state at least twice when processing this suffix. Let z_r^n with $1 \leq n \leq |Q|$ drive M from some state q to state q when processing this suffix. Define φ' to be the word φ with n symbols z_r chopped off. Since φ is accepted we conclude that φ' is accepted as well. However, φ' does not belong to $(L_j)^+$.

Since $L(M) = L_I$, it remains to be shown that φ' does not belong to any of the languages $(L_i)^+$ with $i \in I = \{i_1, i_2, \ldots, i_k\}$ and $i \neq j$.

Assume there is such an i such that $\varphi' \in (L_i)^+$. We consider the structure of all words in $(L_i)^+$. It follows that $x_1 \in \Sigma_D^{(i)}$ and $z_r \in \Sigma_R^{(i)}$.

Case 1. Let there be some $x_t \in \{x_2, x_3, \ldots, x_p\}$ that does not belong to $\Sigma_D^{(i)}$. Since the symbol after x_p is $x_1 \in \Sigma_D^{(i)}$ this implies that a prefix of $x_1 x_2 \cdots x_p$ belongs to $(L_i)^+$. If this prefix is proper, the prefix $(x_1 x_2 \cdots x_p)^2$ of φ' shows that φ' cannot belong to $(L_i)^+$. Therefore, the prefix is not proper and we conclude that the word $x_1 x_2 \cdots x_p$ itself belongs to $(L_i)^+$. However, in this case the prefix $(x_1 x_2 \cdots x_p)^c z_1^{c \cdot p+1} x_1 x_2 \cdots x_p$ of φ' implies that φ' is in a wrong format if $z_1 \in \Sigma_N^{(i)}$ and cannot belong to $(L_i)^+$ if $z_1 \in \Sigma_D^{(i)} \cup \Sigma_R^{(i)}$.

Case 2. We have $\Sigma_D^{(i)} \supseteq \Sigma_D^{(j)}$. Since all symbols $z \in \{z_1, z_2, \ldots, z_{r-1}\}$ have predecessor symbol $x_p \in \Sigma_D^{(i)}$ and successor symbol $x_1 \in \Sigma_D^{(i)}$ in φ', they must belong either to $\Sigma_R^{(i)}$ or to $\Sigma_D^{(i)}$.

If there is some $1 \leq t < r-1$ such that $z_t \in \Sigma_D^{(i)}$ and $z_{t+1} \in \Sigma_R^{(i)}$ then the number $c \cdot p + 1$ of symbols z_{t+1} does not match one plus the number of symbols from $\Sigma_D^{(i)}$ appearing directly before the z_{t+1}. So, φ' does not belong to $(L_i)^+$ in this case. We conclude that if there is some $z_t \in \Sigma_D^{(i)}$, for $1 \leq t \leq r-1$, then all $z \in \{z_t, z_{t+1}, \ldots, z_{r-1}\}$ belong to $\Sigma_D^{(i)}$. Since in the suffix u_r of φ' there are less than $c_r \cdot p + 1$ symbols $z_r \in \Sigma_R^{(i)}$ but $c_r \cdot p$ symbols from $\Sigma_D^{(i)}$, at least one of the symbols from $\{y_1, y_2, \ldots, y_q\}$ must belong to $\Sigma_R^{(i)}$ in order to make φ' belong to $(L_i)^+$. Since any y from $\{y_1, y_2, \ldots, y_q\}$ appears more frequently than the length of its prefix from φ', the word φ' does not belong to $(L_i)^+$ in this case as well. So, we have $\{z_1, z_2, \ldots, z_{r-1}\} \subseteq \Sigma_R^{(i)}$.

Case 3. We have $\Sigma_D^{(i)} \supseteq \Sigma_D^{(j)}$ and $\Sigma_R^{(i)} \supseteq \Sigma_R^{(j)}$. So, the prefix $u_1 u_2 \cdots u_{r-1}$ of φ' belongs to $(L_i)^+$. Now, if there is some $y_t \in \{y_1, y_2, \ldots, y_q\}$ that does not belong to $\Sigma_N^{(i)}$ then a straightforward calculation shows that the suffix $(x_1 x_2 \cdots x_p)^{c_r} y_1^{2 \cdot s} y_2^{2^2 \cdot s} \cdots y_q^{2^q \cdot s} z_r^{c_r \cdot p + 1 - n}$ does not belong to $(L_i)^+$. So, φ' does not belong to $(L_i)^+$ either. This concludes Case 3.

Cases 1 to 3 show that in any case φ' does not belong to $(L_i)^+$. The contradiction implies that φ' does not belong to L_I which in turn is a contradiction to the assumption that L_I is accepted by M. $\square$

4 Closure Properties

Here we are interested in the closure properties of the language families accepted by input-driven counter automata. However, the results for ordinary k-counter automata are complemented by deriving closure under complementation which, in turn, yields non-closure under intersection. The results are summarized in Table 1.

Table 1. Closure properties of the language families discussed. Symbols $\cup_c$, $\cap_c$, and $\cdot_c$ denote union, intersection, and concatenation with compatible signatures. Such operations are not defined for non-input-driven devices and are marked with '—'.

	$\overline{}$	$\cup$	$\cap$	$\cup_c$	$\cap_c$	$\cdot$	$\cdot_c$	$*$	$h_{l.p.}$	REV
REG	yes	yes	yes	—	—	yes	—	yes	yes	yes
$\mathscr{L}$(IDCA(k))	yes	no	no	yes	yes	no	no	no	no	no
$\mathscr{L}$(IDCA)	yes	yes	yes	yes	yes	no	no	no	no	no
$\mathscr{L}$(DCA(k))	yes	no	no	—	—	no	—	no	no	no
$\mathscr{L}$(DCA)	yes	yes	yes	—	—	no	—	no	no	no

We say that two signatures $\Sigma = \Sigma_D \cup \Sigma_R \cup \Sigma_N$ and $\hat{\Sigma} = \hat{\Sigma}_D \cup \hat{\Sigma}_R \cup \hat{\Sigma}_N$ are *compatible* if the symbols shared by both alphabets have the same effect on the counters. That is, if $\bigcup_{j\in\{D,R,N\}}(\Sigma_j \setminus \hat{\Sigma}_j) \cap \hat{\Sigma}$ and $\bigcup_{j\in\{D,R,N\}}(\hat{\Sigma}_j \setminus \Sigma_j) \cap \Sigma$ are empty. Two input-driven counter automata M and M' have compatible signatures, if for any counter of M there is a counter of M' with compatible signature, and vice versa.

Since the devices under consideration are deterministic and are working in realtime, the closure of the accepted language families under complementation can be derived.

Proposition 7. *Let $k \geq 0$. The families of languages accepted by DCA(k), IDCA(k), and IDCA are closed under complementation.*

The property of working input-driven suggests to consider closure properties for the cases where the languages are accepted by devices having compatible signatures.

Proposition 8. *Let $k \geq 0$. The family of languages accepted by IDCA(k) is closed under union and intersection with compatible signatures.*

If we drop the restriction of compatible signatures then multi-counter automata accept language families that are still closed under union and intersection.

Proposition 9. *The family of languages accepted by IDCA is closed under union and intersection.*

We conclude the investigation of closures under Boolean operations by stressing that the restriction to compatible signatures is a serious one. If we drop that restriction for a fixed number of counters, the positive closure property gets lost.

Proposition 10. *Let $k \geq 1$. The family of languages accepted by IDCA(k) is neither closed under union nor under intersection.*

We conclude the section with the investigation of the closure properties under the operations concatenation, iteration, reversal, and length-preserving homomorphism. We use $\{a^m b^n \mid 0 \leq n \leq m\}$, which may leave an unbounded amount of garbage in the counters, as basic witness language to show the non-closures.

Proposition 11. *Let $k \geq 1$. The families of languages accepted by IDCA(k) and IDCA are not closed under concatenation with compatible signatures, iteration, reversal, and length-preserving homomorphism.*

5 Decidability Problems

In this section we turn to explore the decidability problems of emptiness, finiteness, universality, inclusion, equivalence, and regularity for IDCA(k) with $k \geq 0$

counters. Since IDCA(0) are deterministic finite automata, all decidability questions mentioned are decidable. If $k = 1$, we obtain one-counter machines which are special cases of deterministic pushdown automata. Since emptiness, finiteness, universality, equivalence, and regularity is decidable for such automata, we obtain these decidability results for IDCA(1) as well. Finally, the decidability of inclusion for IDCA(1) with compatible signatures is shown in [14]. However, it is shown in [14] as well that the inclusion problem for IDCA(1) becomes undecidable if the signatures are not necessarily compatible.

The complexity of the equivalence problem for DCA(1) is known to be NL-complete [4]. So, it is in NL for IDCA(1) as well. Moreover, since the emptiness problem for deterministic finite automata is already NL-hard, and the emptiness problem easily reduces to the equivalence problem, the latter is NL-complete for IDCA(1).

The inclusion problem for deterministic input-driven pushdown automata with compatible signatures is P-complete [18]. From the NL-hardness of the equivalence problem for IDCA(1) the NL-hardness of the inclusion problem follows. The constructions for the closure under complementation and intersection together with the fact that the emptiness problem for IDCA(1) is in NL yields an NL algorithm for the inclusion problem for IDCA(1). So, the inclusion problem for IDCA(1) is NL-complete.

Now, we consider the case of IDCA with at least two counters. It turns out that in this case undecidability results can be obtained, since it is possible to utilize the undecidability of the halting problem for two-counter machines shown by Minsky [17].

Theorem 12. *Let $k \geq 2$ and M be an IDCA(k). Then it is undecidable whether or not $L(M)$ is empty as well as whether or not $L(M)$ is finite.*

The closure under complementation relates the questions of emptiness and universality, whereas inclusion and equivalence questions can be reduced to universality questions.

Corollary 13. *Let $k \geq 2$ and M be an IDCA(k) over some input alphabet Σ. Then it is undecidable whether or not $L(M) = \Sigma^*$.*

Proof. Due to Proposition 7 it is possible for an IDCA(2) M to construct an IDCA(2) M' accepting the complement $\overline{L(M)}$. Hence, the question of whether $L(M) = \Sigma^*$ is equivalent to $L(M') = \overline{L(M)} = \emptyset$. Since the latter question is undecidable due to Theorem 12 we obtain the claim of the theorem. □

Corollary 14. *Let $k \geq 2$ and M and M' be two IDCA(k) with compatible signatures. Then it is neither decidable whether or not $L(M) \subseteq L(M')$ nor whether or not $L(M) = L(M')$.*

Finally, we obtain that the regularity problem is undecidable as well.

Theorem 15. *Let $k \geq 2$ and M be an IDCA(k). Then it is undecidable whether or not M accepts a regular language.*

The undecidability results obtained obviously hold also for the stronger variants of IDCA as well as for the corresponding non-input-driven counter machines.

References

1. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Babai, L. (ed.) STOC 2004, pp. 202–211. ACM (2004). https://doi.org/10.1145/1007352.1007390
2. Bárány, V., Löding, C., Serre, O.: Regularity problems for visibly pushdown languages. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 420–431. Springer, Heidelberg (2006). https://doi.org/10.1007/11672142_34
3. Bensch, S., Holzer, M., Kutrib, M., Malcher, A.: Input-driven stack automata. In: Baeten, J.C.M., Ball, T., de Boer, F.S. (eds.) TCS 2012. LNCS, vol. 7604, pp. 28–42. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33475-7_3
4. Böhm, S., Göller, S., Jančar, P.: Equivalence of deterministic one-counter automata is NL-complete. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) STOC 2013, pp. 131–140. ACM (2013). https://doi.org/10.1145/2488608.2488626
5. von Braunmühl, B., Verbeek, R.: Input-driven languages are recognized in $\log n$ space. In: Topics in the Theory of Computation, Mathematics Studies, vol. 102, pp. 1–19, North-Holland (1985). https://doi.org/10.1007/3-540-12689-9_92
6. Breveglieri, L., Cherubini, A., Citrini, C., Crespi-Reghizzi, S.: Multi-push-down languages and grammars. Int. J. Found. Comput. Sci. **7**, 253–292 (1996). https://doi.org/10.1142/S0129054196000191
7. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Counter machines and counter languages. Math. Syst. Theory **2**, 265–283 (1968). https://doi.org/10.1007/BF01694011
8. Greibach, S.A.: Remarks on blind and partially blind one-way multicounter machines. Theor. Comput. Sci. **7**, 311–324 (1978). https://doi.org/10.1016/0304-3975(78)90020-8
9. Hahn, M., Krebs, A., Lange, K.-J., Ludwig, M.: Visibly counter languages and the structure of NC^1. In: Italiano, G.F., Pighizzini, G., Sannella, D.T. (eds.) MFCS 2015. LNCS, vol. 9235, pp. 384–394. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48054-0_32
10. Ibarra, O.H.: Reversal-bounded multicounter machines and their decision problems. J. ACM **25**, 116–133 (1978). https://doi.org/10.1145/322047.322058
11. Ibarra, O.H.: Visibly pushdown automata and transducers with counters. Fund. Inform. **148**, 291–308 (2016). https://doi.org/10.3233/FI-2016-1436
12. Krebs, A., Lange, K., Ludwig, M.: Visibly counter languages and constant depth circuits. In: Mayr, E.W., Ollinger, N. (eds.) STACS 2015. LIPIcs, vol. 30, pp. 594–607 (2015). https://doi.org/10.4230/LIPIcs.STACS.2015.594
13. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B., Wendlandt, M.: Deterministic input-driven queue automata: finite turns, decidability, and closure properties. Theor. Comput. Sci. **578**, 58–71 (2015)
14. Kutrib, M., Malcher, A., Wendlandt, M.: Tinput-driven pushdown, counter, and stack automata. Fund. Inform. **155**, 59–88 (2017). https://doi.org/10.3233/FI-2017-1576
15. La Torre, S., Napoli, M., Parlato, G.: Scope-bounded pushdown languages. Int. J. Found. Comput. Sci. **27**, 215–234 (2016)
16. Mehlhorn, K.: Pebbling mountain ranges and its application to DCFL-recognition. In: de Bakker, J., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 422–435. Springer, Heidelberg (1980). https://doi.org/10.1007/3-540-10003-2_89
17. Minsky, M.L.: Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. Ann. Math. **74**, 437–455 (1961). 2nd S
18. Okhotin, A., Salomaa, K.: Complexity of input-driven pushdown automata. SIGACT News **45**, 47–67 (2014). https://doi.org/10.1145/2636805.2636821