Regulated Pushdown Automata

Alexander Meduna and Dusan Kolar

Department of Computer Science and Engineering
Technical University of Brno
Bozetechova 2, Brno 61266, Czech Republic

Abstract

The present paper suggests a new investigation area of the formal language theory—*regulated automata*. Specifically, it investigates pushdown automata that regulate the use of their rules by control languages. It proves that this regulation has no effect on the power of pushdown automata if the control languages are regular. However, the pushdown automata regulated by linear control languages characterize the family of recursively enumerable languages. All these results are established in terms of (A) acceptance by final state, (B) acceptance by empty pushdown, and (C) acceptance by final state and empty pushdown. In its conclusion, this paper formulates several open problems.

Key Words: pushdown automata; regulated accepting; control languages

1 Introduction

Over the past three or four decades, grammars that regulate the use of their rules by various control mechanisms have played an important role in the language theory. Indeed, literally hundreds studies were written about these grammars (see [1], Chapter 5 in the second volume of [4], and Chapter V in [5] for an overview of these studies). Besides grammars, however, the language theory uses automata as fundamental language models, and this very elementary fact gives rise to the idea of regulated automata, which are introduced and discussed in the present paper.

More specifically, this paper introduces pushdown automata that regulate the use of their rules by control languages. First, it demonstrates that this regulation has no effect on the power of pushdown automata if the control languages are regular. Based on this result, it points out that pushdown automata regulated by analogy with the control mechanisms used in most common regulated grammars, such as matrix grammars, are of little interest because their resulting power coincides with the power of ordinary pushdown automata. Then, however, the present paper proves that the pushdown automata increase their power remarkably if they are regulated by linear languages; indeed, they characterize the family of recursively enumerable languages.

All results given in this paper are established in terms of (A) acceptance by final state, (B) acceptance by empty pushdown, and (C) acceptance by final state and empty pushdown. In its conclusion, this paper discusses some open problem areas concerning regulated automata.

2 Preliminaries

We assume that the reader is familiar with the language theory (see [3]). Set $\mathcal{N} = \{1, 2, \ldots\}$ and $\mathcal{I} = \{0, 1, 2, \ldots\}$.

Let V be an alphabet. V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation.

For $w \in V^*$, $|w|$ and $reversal(w)$ denote the length of w and the reversal of w, respectively. Set $prefix(w) = \{x \mid x \text{ is a prefix of } w\}$, $suffix(w) = \{x \mid x \text{ is a suffix of } w\}$, and $alph(w) = \{a \mid a \in V, \text{and } a \text{ appears in } w\}$.

For $w \in V^+$ and $i \in \{1, \ldots, |w|\}$, $sym(w, i)$ denotes the ith symbol of w; for instance, $sym(abcd, 3) = c$.

A *linear grammar* is a quadruple, $G = (N, T, P, S)$, where N and T are alphabets such that $N \cap T = \emptyset$, $S \in N$, and P is a finite set of productions of the form $A \to x$, where $A \in N$ and $x \in T^*(N \cup \{\varepsilon\})T^*$. If $A \to x \in P$ and $u, v \in T^*$, then $uAv \Rightarrow uxv$ $[A \to x]$ or, simply, $uAv \Rightarrow uxv$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$; then, based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of G, $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. A language, L, is *linear* if and only if $L = L(G)$, where G is a linear grammar.

Let $G = (N, T, P, S)$ be a linear grammar. G represents a *regular grammar* if for every $A \to x \in P$, $x \in T(N \cup \{\varepsilon\})$. A language, L, is *regular* if and only if $L = L(G)$, where G is a regular grammar.

A *queue grammar* (see [2]) is a sixtuple, $Q = (V, T, W, F, S, P)$, where V and W are alphabets satisfying $V \cap W = \emptyset$, $T \subseteq V$, $F \subseteq W$, $S \in (V - T)(W - F)$, and $P \subseteq (V \times (W - F)) \times (V^* \times W)$ is a finite relation such that for every $a \in V$, there exists an element $(a, b, x, c) \in P$. If $u, v \in V^*W$ such that $u = arb$, $v = rzc$, $a \in V$, $r, z \in V^*$, $b, c \in W$ and $(a, b, z, c) \in P$, then $u \Rightarrow v$ $[(a, b, z, c)]$ in G or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where $n \geq 0$. Based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as $L(Q) = \{w \in T^* \mid S \Rightarrow^* wf \text{ where } f \in F\}$.

Next, this paper slightly modifies the notion of a queue grammar.

A *left-extended queue grammar* is a sixtuple, $Q = (V, T, W, F, S, P)$, where V, T, W, F, S, P have the same meaning as in a queue grammar; in addition, assume that $\# \notin V \cup W$. If $u, v \in V^*\{\#\}V^*W$ so $u = w\#arb$, $v = wa\#rzc$, $a \in V$, $r, z, w \in V^*$, $b, c \in W$, and $(a, b, z, c) \in P$, then $u \Rightarrow v$ $[(a, b, z, c)]$ in G or, simply, $u \Rightarrow v$. In the standard manner, extend $\Rightarrow$ to $\Rightarrow^n$, where

$n \geq 0$. Based on $\Rightarrow^n$, define $\Rightarrow^+$ and $\Rightarrow^*$. The language of Q, $L(Q)$, is defined as $L(Q) = \{v \in T^* \mid \#S \Rightarrow^* w\#vf \text{ for some } w \in V^* \text{ and } f \in F\}$.

Let *REG*, *LIN*, and *RE* denote the families of regular, linear, and recursively enumerable languages, respectively.

3 Definitions

Consider a pushdown automaton, M, and a control language, Ξ, over M's rules. Informally, with Ξ, M accepts a word, x, if and only if Ξ contains a control word according to which M makes a sequence of moves so it reaches a final configuration after reading x.

Formally, a *pushdown automaton* is a 7-tuple, $M = (Q, \Sigma, \Omega, R, s, S, F)$, where Q is a *finite set of states*, Σ is an *input alphabet*, Ω is a *pushdown alphabet*, R is a *finite set of rules* of the form $Apa \to wq$, where $A \in \Omega$, $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $w \in \Omega^*$, $s \in Q$ is the *start state*, $S \in \Omega$ is the *start symbol*, $F \subseteq Q$ is a *set of final states*. In addition, this paper requires that Q, Σ, Ω are pairwise disjoint.

Let Ψ be an alphabet of *rule labels* such that $card(\Psi) = card(R)$, and ψ be a bijection from R to Ψ. For simplicity, to express that ψ maps a rule, $Apa \to wq \in R$, to ρ, where $\rho \in \Psi$, this paper writes $\rho.Apa \to wq \in R$; in other words, $\rho.Apa \to wq$ means $\psi(Apa \to wq) = \rho$. A *configuration* of M, χ, is any word from $\Omega^* Q \Sigma^*$. For every $x \in \Omega^*$, $y \in \Sigma^*$, and $\rho.Apa \to wq \in R$, M makes a move from configuration $xApay$ to configuration $xwqy$ according to ρ, written as $xApay \Rightarrow xwqy\ [\rho]$. Let χ be any configuration of M. M makes *zero moves from χ to χ according to ε*, symbolically written as $\chi \Rightarrow^0 \chi\ [\varepsilon]$. Let there exist a sequence of configurations $\chi_0, \chi_1, \ldots, \chi_n$ for some $n \geq 1$ such that $\chi_{i-1} \Rightarrow \chi_i\ [\rho_i]$, where $\rho_i \in \Psi$, for $i = 1, \ldots, n$, then M makes *n moves from χ_0 to χ_n according to $\rho_1 \ldots \rho_n$*, symbolically written as $\chi_0 \Rightarrow^n \chi_n\ [\rho_1 \ldots \rho_n]$.

Let Ξ be a *control language* over Ψ; that is, $\Xi \subseteq \Psi^*$. With Ξ, M defines the following three types of accepted languages:

$L(M, \Xi, 1)$—*the language accepted by final state*

$L(M, \Xi, 2)$—*the language accepted by empty pushdown*

$L(M, \Xi, 3)$—*the language accepted by final state and empty pushdown*

defined as follows. Let $\chi \in \Omega^* Q \Sigma^*$. If $\chi \in \Omega^* F$, $\chi \in Q$, $\chi \in F$, then χ is a *1-final configuration*, *2-final configuration*, *3-final configuration*, respectively. For $i = 1, 2, 3$, define $L(M, \Xi, i)$ as $L(M, \Xi, i) = \{w \mid w \in \Sigma^*, \text{ and } Ssw \Rightarrow^* \chi\ [\sigma] \text{ in } M \text{ for an } i\text{-final configuration, } \chi, \text{ and } \sigma \in \Xi\}$.

For any family of languages, X, set $RPD(X, i) = \{L \mid L = L(M, \Xi, i), \text{ where } M \text{ is a pushdown automaton and } \Xi \in X\}$, where $i = 1, 2, 3$. Specifically, $RPD(REG, i)$ and $RPD(LIN, i)$ are central to this paper.

4 Results

This section demonstrates that $CF = RPD(REG, 1) = RPD(REG, 2) = RPD(REG, 3)$ and $RE = RPD(LIN, 1) = RPD(LIN, 2) = RPD(LIN, 3)$.

Some of the following proofs involve several grammars and automata. To avoid any confusion, these proofs sometimes specify a regular grammar, G, as $G = (V[G], P[G], S[G], T[G])$ because this specification clearly expresses that $V[G]$, $P[G]$, $S[G]$, and $T[G]$ represent G's components. Other grammars and automata are specified analogously whenever any confusion may exist.

Regular Control Languages

Next, this section proves that if the control languages are regular, then the regulation of pushdown automata has no effect on their power. The proof of the following lemma presents a transformation that converts any regular grammar, G, and any pushdown automaton, K, to an ordinary pushdown automaton, M, such that $L(M) = L(K, L(G), 1)$.

Lemma 1
For every regular grammar, G, and every pushdown automaton, K, there exists a pushdown automaton, M, such that $L(M) = L(K, L(G), 1)$.

Proof: Let $G = (N[G], T[G], P[G], S[G])$ be any regular grammar, and let $K = (Q[K], \Sigma[K], \Omega[K], R[K], s[K], S[K], F[K])$ be any pushdown automaton. Next, we construct a pushdown automaton, M, that simultaneously simulates G and K so that $L(M) = L(K, L(G), 1)$.

Let f be a new symbol. Define the pushdown automaton $M = (Q[M], \Sigma[M], \Omega[M], R[M], s[M], S[M], F[M])$ as $Q[M] = \{\langle qB \rangle \mid q \in Q[K], B \in N[G] \cup \{f\}\}$, $\Sigma[M] = \Sigma[K]$, $\Omega[M] = \Omega[K]$, $s[M] = \langle s[K]S[G] \rangle$, $S[M] = S[K]$, $F[M] = \{\langle qf \rangle \mid q \in F[K]\}$, and $R[M] = \{C\langle qA \rangle b \rightarrow x\langle pB \rangle \mid a.Cqb \rightarrow xp \in R[K], A \rightarrow aB \in P[G]\} \cup \{C\langle qA \rangle b \rightarrow x\langle pf \rangle \mid a.Cqb \rightarrow xp \in R[K], A \rightarrow a \in P[G]\}$.

Observe that a move in M according to $C\langle qA \rangle b \rightarrow x\langle pB \rangle \in R[M]$ simulates a move in K according $a.Cqb \rightarrow xp \in R[K]$, where a is generated in G by using $A \rightarrow aB \in P[G]$. Based on this observation, it is rather easy to see that M accepts an input word, w, if and only if K reads w and enters a final state after using a complete word of $L(G)$; therefore, $L(M) = L(K, L(G), 1)$. A rigorous proof that $L(M) = L(K, L(G), 1)$ is left to the reader. ■

Theorem 2
For $i \in \{1, 2, 3\}$, $CF = RPD(REG, i)$.

Proof: To prove $CF = RPD(REG, 1)$, notice that $RPD(REG, 1) \subseteq CF$ follows from Lemma 1. Clearly, $CF \subseteq RPD(REG, 1)$, so $RPD(REG, 1) = CF$.

By analogy with the demonstration of $RPD(REG,1) = CF$, prove that $CF = RPD(REG,2)$ and $CF = RPD(REG,3)$. ■

Let us point out that most fundamental regulated grammars use control mechanisms that can be expressed in terms of regular control languages (c.f. Theorem V.6.1 on page 175 in [5]). However, pushdown automata introduced by analogy with these grammars are of little or no interest because they are as powerful as ordinary pushdown automata (see Theorem 2 above).

Linear Control Languages

The rest of this section demonstrates that the pushdown automata regulated by linear control languages are more powerful than ordinary pushdown automata. In fact, it proves that $RE = RPD(LIN,1) = RPD(LIN,2) = RPD(LIN,3)$.

Lemma 3
For every left-extended queue grammar, K, there exists a left-extended queue grammar $Q = (V, T, W, F, s, P)$ satisfying $L(K) = L(Q)$, $!$ is a distinguished member of $(W - F)$, $V = U \cup Z \cup T$ such that U, Z, T are pairwise disjoint, and Q derives every $z \in L(Q)$ in this way

$$\begin{array}{lll}
\#S & \Rightarrow^+ & x\#b_1b_2\ldots b_n! \\
 & \Rightarrow & xb_1\#b_2\ldots b_n y_1 p_2 \\
 & \Rightarrow & xb_1b_2\#b_3\ldots b_n y_1y_2p_3 \\
 & \vdots & \\
 & \Rightarrow & xb_1b_2\ldots b_{n-1}\#b_n y_1y_2\ldots y_{n-1}p_n \\
 & \Rightarrow & xb_1b_2\ldots b_{n-1}b_n\#y_1y_2\ldots y_np_{n+1}
\end{array}$$

where $n \in N$, $x \in U^*$, $b_i \in Z$ for $i = 1, \ldots, n$, $y_i \in T^*$ for $i = 1, \ldots, n$, $z = y_1y_2\ldots y_n$, $p_i \in W - \{!\}$ for $i = 1, \ldots, n-1$, $p_n \in F$, and in this derivation $x\#b_1b_2\ldots b_n!$ is the only word containing $!$.

Proof: Let K be any left-extended queue grammar. Convert K to a left-extended queue grammar, $H = (V[H], T[H], W[H], F[H], S[H], P[H])$, such that $L(K) = L(H)$ and H generates every $x \in L(H)$ by making two or more derivation steps (this conversion is trivial and left to the reader).

Define the bijection α from W to W', where $W' = \{q' \mid q \in W\}$, as $\alpha(q) = \{q'\}$ for every $q \in W$. Analogously, define the bijection β from W to W'', where $W'' = \{q'' \mid q \in W\}$, as $\beta(q) = \{q''\}$ for every $q \in W$. Without any loss of generality, assume that $\{1,2\} \cap (V \cup W) = \emptyset$. Set $\Xi = \{\langle a, q, u1v, p\rangle \mid (a, q, uv, p) \in P[H]$ for some $a \in V, q \in W - F, v \in T^*, u \in V^*$, and $p \in W\}$ and $\Gamma = \{\langle a, q, z2w, p\rangle \mid (a, q, zw, p) \in P[H]$ for some $a \in V, q \in W - F, w \in T^*, z \in V^*$, and $p \in W\}$. Define the relation χ from $V[H]$ to $\Xi\Gamma$ so for every $a \in V$, $\chi(a) = \{\langle a, q, y1x, p\rangle\langle a, q, y2x, p\rangle \mid \langle a, q, y1x, p\rangle \in \Xi, \langle a, q, y2x, p\rangle \in \Gamma, q \in W -$

$F, x \in T^*, y \in V^*, p \in W\}$. Define the bijection δ from $V[H]$ to V', where $V' = \{a' \mid a \in V\}$, as $\delta(a) = \{a'\}$. In the standard manner, extend δ so it is defined from $(V[H])^*$ to $(V')^*$. Finally, define the bijection ϕ from $V[H]$ to V'', where $V'' = \{a'' \mid a \in V\}$, as $\phi(a) = \{a''\}$. In the standard manner, extend ϕ so it is defined from $(V[H])^*$ to $(V'')^*$.

Define the left-extended queue grammar

$$Q = (V[Q], T[Q], W[Q], F[Q], S[Q], P[Q])$$

so that $V[Q] = V[H] \cup \delta(V[H]) \cup \phi(V[H]) \cup \Xi \cup \Gamma$, $T[Q] = T[H]$, $W[Q] = W[H] \cup \alpha(W[H]) \cup \beta(W[H]) \cup \{!\}$, $F[Q] = \beta(F[H])$, $S[Q] = \delta(S[H])$, and $P[V]$ is constructed in this way

1. if $(a, q, x, p) \in P[H]$ where $a \in V$, $q \in W - F$, $x \in V^*$, and $p \in W$, then add $(\delta(a), q, \delta(x), p)$ and $(\delta(a), \alpha(q), \delta(x), \alpha(p))$ to $P[Q]$;
2. if $(a, q, xAy, p) \in P[H]$, where $a \in V$, $q \in W - F$, $x, y \in V^*$, $A \in V$, and $p \in W$, then add $(\delta(a), q, \delta(x)\chi(A)\phi(y), \alpha(p))$ to $P[Q]$;
3. if $(a, q, yx, p) \in P[H]$, where $a \in V$, $q \in W - F$, $y \in V^*$, $x \in T^*$, and $p \in W$, then add $(\langle a, q, y1x, p\rangle, \alpha(q), \phi(y), !)$ and $(\langle a, q, y2x, p\rangle, !, x, \beta(p))$ to $P[Q]$;
4. if $(a, q, y, p) \in P[H]$, where $a \in V$, $q \in W - F$, $y \in T^*$, and $p \in W$, then add $(\phi(a), \beta(q), y, \beta(p))$ to $P[Q]$.

Set $U = \delta(V[H]) \cup \Xi$ and $Z = \phi(V[H]) \cup \Gamma$. Notice that Q satisfies properties 2 and 3 of Lemma 3. To demonstrate that the other two properties hold as well, observe that H generates every $z \in L(H)$ in this way

$$\begin{array}{lll}
\#S[H] & \Rightarrow^+ & x\#b_1b_2\dots b_ip_1 \\
 & \Rightarrow & xb_1\#b_2\dots b_ib_{i+1}\dots b_ny_1p_2 \\
 & \Rightarrow & xb_1b_2\#b_3\dots b_ib_{i+1}\dots b_ny_1y_2p_3 \\
 & \vdots & \\
 & \Rightarrow & xb_1b_2\dots b_{i-1}\#b_ib_{i+1}\dots b_ny_1y_2\dots y_{i-1}p_i \\
 & \Rightarrow & xb_1b_2\dots b_i\#b_{i+1}\dots b_ny_1y_2\dots y_{i-1}y_ip_{i+1} \\
 & \vdots & \\
 & \Rightarrow & xb_1b_2\dots b_{n-1}\#b_ny_1y_2\dots y_{n-1}p_n \\
 & \Rightarrow & xb_1b_2\dots b_{n-1}b_n\#y_1y_2\dots y_np_{n+1}
\end{array}$$

where $n \in \mathcal{N}$, $x \in V^+$, $b_i \in V$ for $i = 1, \dots, n$, $y_i \in T^*$ for $i = 1, \dots, n$, $z = y_1y_2\dots y_n$, $p_i \in W$ for $i = 1, \in, n$, $p_{n+1} \in F$. Q simulates this generation of z as follows

$$
\begin{array}{lll}
\#S[Q] & \Rightarrow^+ & \delta(x)\#\chi(b_1)\phi(b_2\ldots b_i)\alpha(p_1) \\
 & \Rightarrow & \delta(x)\langle b_1, p_1, b_{i+1}\ldots b_n 1y_1, p_2\rangle\#\langle b_1, p_1, b_{i+1}\ldots b_n 2y_1, p2\rangle \\
 & & \phi(b_2\ldots b_i b_{i+1}\ldots b_n)! \\
 & \Rightarrow & \delta(x)\chi(b_1)\#\phi(b_2\ldots b_n)y_1 p_2 \\
 & \Rightarrow & \delta(x)\chi(b_1)\phi(b_2)\#\phi(b_3\ldots b_n)y_1 y_2 p_3 \\
 & \vdots & \\
 & \Rightarrow & \delta(x)\chi(b_1)\phi(b_2\ldots b_{n-1})\#\phi(b_n)y_1 y_2\ldots y_{n-1}p_n \\
 & \Rightarrow & \delta(x)\chi(b_1)\phi(b_2\ldots b_n)\#y_1 y_2\ldots y_n p_{n+1}
\end{array}
$$

Q makes the first $|x|-1$ steps of $\#S[Q] \Rightarrow^+ \delta(x)\#\chi(b_1)\phi(b_2\ldots b_i)\alpha(p_1)$ according to productions introduced in 1; in addition, during this derivation, Q makes one step by using a production introduced in 2. By using productions introduced in 3, Q makes the two steps

$$
\begin{array}{lr}
\delta(x)\#\chi(b_1)\phi(b_2\ldots b_i)\alpha(p_0) & \Rightarrow \\
\delta(x)\langle b_1, p_1, b_{i+1}\ldots b_n 1y_1, p_2\rangle\#\langle b_1, p_1, b_{i+1}\ldots b_n 2y_1, p_2\rangle\phi(b_2\ldots b_i b_{i+1}\ldots b_n)! & \Rightarrow \\
\delta(x)\chi(b_1)\#\phi(b_2\ldots b_n)y_1 p_2 &
\end{array}
$$

with

$$\chi(b_1) = \langle b_1, p_0, b_{i+1}\ldots b_n 1y_1, p_1\rangle\langle b_1, p_0, b_{i+1}\ldots b_n 2y_1, p2\rangle.$$

Q makes the rest of the derivation by using productions introduced in 4.

Based on the previous observation, it easy to see that Q satisfies all the four properties stated in Lemma 3, whose rigorous proof is left to the reader. ■

Lemma 4

Let Q be a left-extended queue grammar that satisfies the properties of Lemma 3. Then, there exist a linear grammar, G, and a pushdown automaton, M, such that $L(Q) = L(M, L(G), 3)$.

Proof: Let $Q = (V[Q], T[Q], W[Q], F[Q], s[Q], P[Q])$ be a left-extended queue grammar satisfying the properties of Lemma 3. Without any loss of generality, assume that $\{@, £, ¶\} \cap (V \cup W) = \emptyset$. Define the coding, ζ, from $(V[Q])^*$ to $\{\langle £as\rangle \mid a \in V[Q]\}^*$ as $\zeta(a) = \{\langle £as\rangle\}$ (s is used as the start state of the pushdown automaton, M, defined later in this proof).

Construct the linear grammar $G = (N[G], T[G], P[G], S[G])$ in the following way. Initially, set

$$N[G] = \{S[G], \langle !\rangle, \langle !, 1\rangle\} \cup \{\langle f\rangle \mid f \in F[Q]\}$$

$$T[G] = \zeta(V[Q]) \cup \{\langle £\S s\rangle, \langle £@\rangle\} \cup \{\langle £\S f\rangle \mid f \in F[Q]\}$$

$$P[G] = \{S[G] \rightarrow \langle £\S s\rangle\langle f\rangle \mid f \in F[Q]\} \cup \{\langle !\rangle \rightarrow \langle !, \mathit{1}\rangle\langle £@\rangle\}$$

Increase $N[G]$, $T[G]$, and $P[G]$ by performing 1 through 3, following next.

1. for every $(a,p,x,q) \in P[Q]$ where $p,q \in W[Q]$, $a \in Z$, $x \in T^*$,

$$\begin{aligned}
N[G] &= N[G] \cup \{\langle apxqk\rangle \mid k = 0,\ldots,|x|\} \cup \{\langle p\rangle, \langle q\rangle\}\\
T[G] &= T[G] \ \cup \{\langle \pounds\mathrm{sym}(y,k)\rangle \mid k = \mathit{1},\ldots,|y|\} \cup \{\langle \pounds\, apxq\rangle\}\\
P[G] &= P[G] \ \cup \{\langle q\rangle \to \langle apxq|x|\rangle\langle \pounds\, apxq\rangle, \langle apxq\mathit{0}\rangle \to \langle p\rangle\}\\
&\qquad\quad \cup \{\langle apxqk\rangle \to \langle apxq(k-1)\rangle\langle \pounds\mathrm{sym}(x,k)\rangle \mid k = \mathit{1},\ldots,|x|\};
\end{aligned}$$

2. for every $(a,p,x,q) \in P[Q]$ with $p,q \in W[Q]$, $a \in U$, $x \in (V[Q])^*$,

$$\begin{aligned}
N[G] &= N[G] \cup \{\langle p,1\rangle, \langle q,1\rangle\}\\
P[G] &= P[G] \ \cup \{\langle q,1\rangle \to \mathrm{reversal}(\zeta(x))\langle p,1\rangle\zeta(a)\};
\end{aligned}$$

3. for every $(a,p,x,q) \in P[Q]$ with $ap = S[Q]$, $p,q \in W[Q]$, $x \in (V[Q])^*$,

$$\begin{aligned}
N[G] &= N[G] \cup \{\langle q,1\rangle\}\\
P[G] &= P[G] \ \cup \{\langle q,1\rangle \to \mathrm{reversal}(x)\langle \pounds\$s\rangle\}.
\end{aligned}$$

The construction of G is completed. Set $\Psi = T[G]$. Ψ represents the alphabet of rule labels corresponding to the rules of the pushdown automaton $M = (Q[M], \Sigma[M], \Omega[M], R[M], s[M], S[M], \{\rceil\})$, which is constructed next.

Initially, set $Q[M] = \{s[M], \langle\P!\rangle, \lfloor, \rceil\}$ (throughout the rest of this proof, $s[M]$ is abbreviated to s), $\Sigma[M] = T[Q]$, $\Omega[M] = \{S[M], \S\} \cup V[Q]$, $R[M] = \{\langle \pounds\S s\rangle.S[M]s \to \S s\} \cup \{\langle \pounds\S f\rangle.\S\langle\P f\rangle \to\rceil \mid f \in F[M]\}$. Increase $Q[M]$ and $R[M]$ by performing A through D, following next.

A. $R[M] = R[M] \cup \{\langle \pounds\, bs\rangle.as \to abs \mid a \in \mathit{\Omega}[M] - \{S[M]\}, b \in \mathit{\Omega}[M] - \{\$\}\}$;

B. $R[M] = R[M] \cup \{\langle \pounds\$s\rangle.as \to a\lfloor \mid a \in V[Q]\} \cup \{\langle \pounds\, a\rangle.a\lfloor\to \lfloor \mid a \in V[Q]\}$;

C. $R[M] = R[M] \cup \{\langle \pounds@\rangle.a\lfloor\to a\langle\P!\rangle \mid a \in Z\}$;

D. for every $(a,p,x,q) \in P[Q]$, where $p,q \in W[Q]$, $a \in Z$, $x \in (T[Q])^*$,

$$\begin{aligned}
Q[M] &= Q[M] \cup \{\langle\P p\rangle\} \cup \{\langle\P qu\rangle \mid u \in \mathrm{prefix}(x)\}\\
R[M] &= R[M] \cup \{\langle \pounds\, b\rangle.a\langle\P qy\rangle b \to a\langle\P qyb\rangle \mid b \in T[Q], y \in (T[Q])^*,\\
&\qquad yb \in \mathrm{prefix(x)}\} \cup \{\langle \pounds\, apxq\rangle.a\langle\P qx\rangle \to \langle\P p\rangle\}.
\end{aligned}$$

The construction of M is completed.

Notice that several components of G and M have this form: $\langle x\rangle$. Intuitively, if x begins with $\pounds$, then $\langle x\rangle \in T[G]$. If x begins with $\P$, then $\langle x\rangle \in Q[M]$. Finally, if x begins with a symbol different from $\pounds$ or $\P$, then $\langle x\rangle \in N[G]$.

First, we only sketch the reason why $L(Q)$ contains $L(M, L(G), 3)$. Accordinng to a word from $L(G)$, M accepts every word w as

$$
\begin{array}{lll}
\S w_1 \dots w_{m-1} w_m & \Rightarrow^+ & \S b_m \dots b_1 a_n \dots a_1 s w_1 \dots w_{m-1} w_m \\
& \Rightarrow & \S b_m \dots b_1 a_n \dots a_1 \lfloor w_1 \dots w_{m-1} w_m \\
& \Rightarrow^n & \S b_m \dots b_1 \lfloor w_1 \dots w_{m-1} w_m \\
& \Rightarrow & \S b_m \dots b_1 \langle \P q_1 \rangle w_1 \dots w_{m-1} w_m \\
& \Rightarrow^{|w_1|} & \S b_m \dots b_1 \langle \P q_1 w_1 \rangle w_2 \dots w_{m-1} w_m \\
& \Rightarrow & \S b_m \dots b_2 \langle \P q_2 \rangle w_2 \dots w_{m-1} w_m \\
& \Rightarrow^{|w_2|} & \S b_m \dots b_2 \langle \P q_2 w_2 \rangle w_3 \dots w_{m-1} w_m \\
& \Rightarrow & \S b_m \dots b_3 \langle \P q_3 \rangle w_3 \dots w_{m-1} w_m \\
& \vdots & \\
& \Rightarrow & \S b_m \langle \P q_m \rangle w_m \\
& \Rightarrow^{|w_m|} & \S b_m \langle \P q_m w_m \rangle \\
& \Rightarrow & \S \langle \P q_{m+1} \rangle \\
& \Rightarrow & \rceil
\end{array}
$$

where $w = w_1 \dots w_{m-1} w_m$, $a_1 \dots a_n b_1 \dots b_m = x_1 \dots x_{n+1}$, and $R[Q]$ contains $(a_0, p_0, x_1, p_1), (a_1, p_1, x_2, p_2), \dots, (a_n, p_n, x_{n+1}, q_1), (b_1, q_1, w_1, q_2), (b_2, q_2, w_2, q_3), \dots, (b_m, q_m, w_m, q_{m+1})$. According to these members of $R[Q]$, Q makes

$$
\begin{array}{llll}
\# a_0 p_0 & \Rightarrow & a_0 \# y_0 x_1 p_1 & [(a_0, p_0, x_1, p_1)] \\
& \Rightarrow & a_0 a_1 \# y_1 x_2 p_2 & [(a_1, p_1, x_2, p_2)] \\
& \Rightarrow & a_0 a_1 a_2 \# y_2 x_3 p_3 & [(a_2, p_2, x_3, p_3)] \\
& \vdots & & \\
& \Rightarrow & a_0 a_1 a_2 \dots a_{n-1} \# y_{n-1} x_n p_n & [(a_{n-1}, p_{n-1}, x_n, p_n)] \\
& \Rightarrow & a_0 a_1 a_2 \dots a_n \# y_n x_{n+1} q_1 & [(a_n, p_n, x_{n+1}, q_1)] \\
& \Rightarrow & a_0 \dots a_n b_1 \# b_2 \dots b_m w_1 q_2 & [(b_1, q_1, w_1, q_2)] \\
& \Rightarrow & a_0 \dots a_n b_1 b_2 \# b_3 \dots b_m w_1 w_2 q_3 & [(b_2, q_2, w_2, q_3)] \\
& \vdots & & \\
& \Rightarrow & a_0 \dots a_n b_1 \dots b_{m-1} \# b_m w_1 w_2 \dots w_{m-1} q_m & [(b_{m-1}, q_{m-1}, w_{m-1}, q_m)] \\
& \Rightarrow & a_0 \dots a_n b_1 \dots b_m \# w_1 w_2 \dots w_m q_{m+1} & [(b_m, q_m, w_m, q_{m+1})]
\end{array}
$$

Therefore, $L(M, L(G), 3) \subseteq L(Q)$.

More formally, to demonstrate that $L(Q)$ contains $L(M, L(G), 3)$, consider any $h \in L(G)$. G generates h as

$$
\begin{array}{ll}
S[G] \Rightarrow & \langle £\S s \rangle \langle q_{m+1} \rangle \\
\Rightarrow^{|w_m|+1} & \langle £\S s \rangle \langle q_m \rangle t_m \langle £\, b_m\, q_m\, w_m\, q_{m+1} \rangle \\
\Rightarrow^{|w_{m-1}|+1} & \langle £\S s \rangle \langle q_{m-1} \rangle t_{m-1} \langle £\, b_{m-1}\, q_{m-1}\, w_{m-1}\, q_m \rangle t_m \langle £\, b_m\, q_m\, w_m\, q_{m+1} \rangle \\
\vdots & \\
\Rightarrow^{|w_1|+1} & \langle £\S s \rangle \langle q_1 \rangle o \\
\Rightarrow^{|w_1|+1} & \langle £\S s \rangle \langle q_1, 1 \rangle \langle £@ \rangle o \\
& [\langle q_1 \rangle \rightarrow \langle q_1, 1 \rangle \langle £@ \rangle] \\
& \\
\Rightarrow & \langle £\S s \rangle \zeta(\text{reversal}(x_{n+1})) \langle p_n, 1 \rangle \langle £\, a_n \rangle \langle £@ \rangle o
\end{array}
$$

$$[\langle q_1, 1\rangle \to \text{reversal}(\zeta(x_{n+1}))\langle p_n, 1\rangle\langle £\, a_n\rangle\langle £@\rangle]$$

$$\begin{array}{ll} \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_n x_{n+1}))\langle p_{n-1}, 1\rangle\langle £\, a_{n-1}\rangle\langle £\, a_n\rangle\langle £@\rangle o \\ & [\langle p_n, 1\rangle \to \text{reversal}(\zeta(x_n))\langle p_{n-1}, 1\rangle\langle £\, a_{n-1}\rangle] \\ \vdots & \\ \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_2 \dots x_n x_{n+1}))\langle p1, 1\rangle\langle £\, a_1\rangle\langle £\, a_2\rangle \dots \langle £\, a_n\rangle\langle £@\rangle o \\ & [\langle p_2, 1\rangle \to \text{reversal}(\zeta(x_2))\langle p_1, 1\rangle\langle £\, a_1\rangle] \\ \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_1 \dots x_n x_{n+1}))\langle £\$ s\rangle\langle £\, a_1\rangle\langle £\, a_2\rangle \dots \langle £\, a_n\rangle\langle £@\rangle o \\ & [\langle p_1, 1\rangle \to \text{reversal}(\zeta(x_1))\langle £\$ s\rangle] \end{array}$$

where $n, m \in \mathcal{N}$; $a_i \in U$ for $i = 1, \dots, n$; $b_k \in Z$ for $k = 1, \dots, m$; $x_l \in V^*$ for $l = 1, \dots, n+1$; $p_i \in W$ for $i = 1, \dots, n$; $q_l \in W$ for $l = 1, \dots, m+1$ with $q_1 = !$ and $q_{m+1} \in F$; $t_k = \langle £\text{sym}(w_k, 1)\rangle \dots \langle £\text{sym}(w_k, |w_k| - 1)\rangle\langle £\text{sym}(w_k, |w_k|)\rangle$ for $k = 1, \dots, m$; $o = t_1\langle £\, b_1\, q_1\, w_1\, q_2\rangle \dots \langle £\S s\rangle\langle q_{m-1}\rangle t_{m-1}\langle £\, b_{m-1}\, q_{m-1}\, w_{m-1}\, q_m\rangle t_m\langle £\, b_m\, q_m\, w_m\, q_{m+1}\rangle$; $h = \langle £\S s\rangle\zeta(\text{reversal}(x_1 \dots x_n x_{n+1}))\langle £\$\rangle\langle £\, a_1\rangle\langle £\, a_2\rangle \dots \langle £\, a_n\rangle\langle £@\rangle o$.

In greater detail, G makes $S[G] \Rightarrow \langle £\S s\rangle\langle q_{m+1}\rangle$ according to $S[G] \to \langle £\S s\rangle\langle q_{m+1}\rangle$. Furthermore, G makes

$$\begin{array}{ll} & \langle £\S s\rangle\langle q_{m+1}\rangle \\ \Rightarrow^{|w_m|+1} & \langle £\S s\rangle\langle q_m\rangle t_m\langle £\, b_m q_m w_m q_{m+1}\rangle \\ \Rightarrow^{|w_{m-1}|+1} & \langle £\S s\rangle\langle q_{m-1}\rangle t_{m-1}\langle £\, b_{m-1}\, q_{m-1}\, w_{m-1}\, q_m\rangle t_m\langle £\, b_m q_m w_m q_{m+1}\rangle \\ \vdots & \\ \Rightarrow^{|w_1|+1} & \langle £\S s\rangle\langle q_1\rangle o \end{array}$$

according to productions introduced in step 1. Then, G makes

$$\langle £\S s\rangle\langle q_1\rangle o \Rightarrow \langle £\S s\rangle\langle q_1, 1\rangle\langle £@\rangle o$$

according to $\langle !\rangle \to \langle !, 1\rangle\langle £@\rangle$ (recall that $q_1 = !$). After this step, G makes

$$\begin{array}{ll} & \langle £\S s\rangle\langle q_1, 1\rangle\langle £@\rangle o \\ \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_{n+1}))\langle p_n, 1\rangle\langle £\, a_n\rangle\langle £@\rangle o \\ \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_n x_{n+1}))\langle p_{n-1}, 1\rangle\langle £\, a_{n-1}\rangle\langle £\, a_n\rangle\langle £@\rangle o \\ \vdots & \\ \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_2 \dots x_n x_{n+1}))\langle p_1, 1\rangle\langle £\, a_1\rangle\langle £\, a_2\rangle \dots \langle £\, a_n\rangle\langle £@\rangle o \end{array}$$

according to productions introduced in step 2. Finally, according to $\langle p_1, 1\rangle \to \text{reversal}(\zeta(x_1))\langle £\$\rangle$, which is introduced in step 3, G makes

$$\begin{array}{ll} & \langle £\S s\rangle\zeta(\text{reversal}(x_2 \dots x_n x_{n+1}))\langle p_1, 1\rangle\langle £\, a_1\rangle\langle £\, a_2\rangle \dots \langle £\, a_n\rangle\langle £@\rangle o \\ \Rightarrow & \langle £\S s\rangle\zeta(\text{reversal}(x_1 \dots x_n x_{n+1}))\langle £\$\rangle\langle £\, a_1\rangle\langle £\, a_2\rangle \dots \langle £\, a_n\rangle\langle £@\rangle o \end{array}$$

If $a_1 \ldots a_n b_1 \ldots b_m$ differs from $x_1 \ldots x_{n+1}$, then M does not accept according to h. Assume that $a_1 \ldots a_n b_1 \ldots b_m = x_1 \ldots x_{n+1}$. At this point, according to h, M makes this sequence of moves

$$
\begin{array}{lll}
\S w_1 \ldots w_{m-1} w_m & \Rightarrow^+ & \S b_m \ldots b_1 a_n \ldots a_1 s w_1 \ldots w_{m-1} w_m \\
& \Rightarrow & \S b_m \ldots b_1 a_n \ldots a_1 \lfloor w_1 \ldots w_{m-1} w_m \\
& \Rightarrow^n & \S b_m \ldots b_1 \lfloor w_1 \ldots w_{m-1} w_m \\
& \Rightarrow & \S b_m \ldots b_1 \langle \P q_1 \rangle w_1 \ldots w_{m-1} w_m \\
& \Rightarrow^{|w_1|} & \S b_m \ldots b_1 \langle \P q_1 w_1 \rangle w_2 \ldots w_{m-1} w_m \\
& \Rightarrow & \S b_m \ldots b_2 \langle \P q_2 \rangle w_2 \ldots w_{m-1} w_m \\
& \Rightarrow^{|w_2|} & \S b_m \ldots b_2 \langle \P q_2 w_2 \rangle w_3 \ldots w_{m-1} w_m \\
& \Rightarrow & \S b_m \ldots b_3 \langle \P q_3 \rangle w_3 \ldots w_{m-1} w_m \\
& \vdots & \\
& \Rightarrow & \S b_m \langle \P q_m \rangle w_m \\
& \Rightarrow^{|w_m|} & \S b_m \langle \P q_m w_m \rangle \\
& \Rightarrow & \S \langle \P q_{m+1} \rangle \\
& \Rightarrow & \rceil
\end{array}
$$

In other words, according to h, M accepts $w_1 \ldots w_{m-1} w_m$. Return to the generation of h in G. By the construction of $P[G]$, this generation implies that $R[Q]$ contains $(a_0, p_0, x_1, p_1), (a_1, p_1, x_2, p_2), \ldots, (a_{j-1}, p_{j-1}, x_j, p_j), \ldots, (a_n, p_n, x_{n+1}, q_1), (b_1, q_1, w_1, q_2), (b_2, q_2, w_2, q_3), \ldots, (b_m, q_m, w_m, q_{m+1})$.

Thus, in Q,

$$
\begin{array}{lll l}
\# a_0 p_0 & \Rightarrow & a_0 \# y_0 x_1 p_1 & [(a_0, p_0, x_1, p_1)] \\
& \Rightarrow & a_0 a_1 \# y_1 x_2 p_2 & [(a_1, p_1, x_2, p_2)] \\
& \Rightarrow & a_0 a_1 a_2 \# y_2 x_3 p_3 & [(a_2, p_2, x_3, p_3)] \\
& \vdots & & \\
& \Rightarrow & a_0 a_1 a_2 \ldots a_{n-1} \# y_{n-1} x_n p_n & [(a_{n-1}, p_{n-1}, x_n, p_n)] \\
& \Rightarrow & a_0 a_1 a_2 \ldots a_n \# y_n x_{n+1} q_1 & [(a_n, p_n, x_{n+1}, q_1)] \\
& \Rightarrow & a_0 \ldots a_n b_1 \# b_2 \ldots b_m w_1 q_2 & [(b_1, q_1, w_1, q_2)] \\
& \Rightarrow & a_0 \ldots a_n b_1 b_2 \# b_3 \ldots b_m w_1 w_2 q_3 & [(b_2, q_2, w_2, q_3)] \\
& \vdots & & \\
& \Rightarrow & a_0 \ldots a_n b_1 \ldots b_{m-1} \# b_m w_1 w_2 \ldots w_{m-1} q_m & [(b_{m-1}, q_{m-1}, w_{m-1}, q_m)] \\
& \Rightarrow & a_0 \ldots a_n b_1 \ldots b_m \# w_1 w_2 \ldots w_m q_{m+1} & [(b_m, q_m, w_m, q_{m+1})]
\end{array}
$$

Therefore, $w_1 w_2 \ldots w_m \in L(Q)$. Consequently, $L(M, L(G), 3) \subseteq L(Q)$.

A proof that that $L(Q) \subseteq L(M, L(G), 3)$ is left to the reader. As $L(Q) \subseteq L(M, L(G), 3)$ and $L(M, L(G), 3) \subseteq L(Q)$, $L(Q) = L(M, L(G), 3)$. Therefore, Lemma 4 holds. ∎

Theorem 5
For $i \in \{1, 2, 3\}$, $RE = RPD(LIN, i)$.

Proof: Obviously, $RPD(LIN, 3) \subseteq RE$. To prove $RE \subseteq RPD(LIN, 3)$, consider any recursively enumerable language, $L \in RE$. By Theorem 2.1 in [2], $L(Q) = L$, for a queue grammar. Clearly, there exists a left-extended queue grammar, Q', so $L(Q) = L(Q')$. Furthermore, by Lemmas 3 and 4, $L(Q') = L(M, L(G), 3)$, for a linear grammar, G, and a pushdown automaton, M. Thus, $L = L(M, L(G), 3)$. Hence, $RE \subseteq RPD(LIN, 3)$. As $RPD(LIN, 3) \subseteq RE$ and $RE \subseteq RPD(LIN, 3)$, $RE = RPD(LIN, 3)$.

By analogy with the demonstration of $RE = RPD(LIN, 3)$, prove $RE = RPD(LIN, i)$ for $i = 1, 2$. ∎

5 Future Investigation

This section suggests some open problem areas concerning regulated automata. First, it states four new areas of investigation. Then, it makes a note about a relation of regulated automata to the classical formal language theory.

New investigation areas

A. For $i = 1, \ldots, 3$, consider $RPD(X, i)$, where X is a language family satisfying $REG \subset X \subset LIN$; for instance, set X equal to the family of minimal linear languages. Compare RE with $RPD(X, i)$.

B. Investigate special cases of regulated pushdown automata, such as their deterministic versions.

C. By analogy with regulated pushdown automata, introduce and study some other types of regulated automata.

D. Investigate the descriptional complexity of regulated pushdown automata.

Classical investigation areas

As already pointed out, this paper has discussed regulated automata as a new investigation field of the formal language theory. Therefore, it has defined all notions and established all results in terms of this new field. However, this approach does not rule out a relation of the achieved results to the classical formal language theory. Specifically, Theorem 5 can be viewed as a new characterization of RE and compared with other well-known characterizations of this family (see pages 180 through 184 in the first volume of [4] for an overview of these characterizations).

References

[1] Dassow, J. and Paun, G.: *Regulated Rewriting in Formal Language Theory*. Springer, New York, 1989.

[2] Kleijn, H. C. M. and Rozenberg, G.: On the Generative Power of Regular Pattern Grammars, *Acta Informatica*, Vol. 20, pp. 391–411, 1983.

[3] Meduna, A.: *Automata and Languages: Theory and Applications*. Springer, London, 2000.

[4] Rozenberg, G. and Salomaa, A. (eds.): *Handbook of Formal Languages; Volumes 1 through 3*. Springer, 1997.

[5] Salomaa, A.: *Formal Languages*. Academic Press, 1973.