Permutations generated by a depth 2 stack and an infinite stack in series are algebraic

Murray Elder   Geoffrey Lee

School of Mathematical and Physical Sciences
The University of Newcastle
Callaghan, New South Wales, 2308, Australia

murray.elder@newcastle.edu.au   geoffrey.a.lee@uon.edu.au

Andrew Rechnitzer

Department of Mathematics
University of British Columbia
Vancouver, British Columbia, V6T-1Z2, Canada

andrewr@math.ubc.ca

Submitted: Aug 1, 2014; Accepted: Apr 22, 2015; Published: Apr 29, 2015
Mathematics Subject Classifications: 05A05

Abstract

We prove that the class of permutations generated by passing an ordered sequence $12\dots n$ through a stack of depth 2 and an infinite stack in series is in bijection with an unambiguous context-free language, where a permutation of length n is encoded by a string of length $3n$. It follows that the sequence counting the number of permutations of each length has an algebraic generating function. We use the explicit context-free grammar to compute the generating function:

$$\sum_{n\geqslant 0} c_n t^n = \frac{(1+q)\left(1+5q-q^2-q^3-(1-q)\sqrt{(1-q^2)(1-4q-q^2)}\right)}{8q}$$

where c_n is the number of permutations of length n that can be generated, and $q \equiv q(t) = \frac{1-2t-\sqrt{1-4t}}{2t}$ is a simple variant of the Catalan generating function. This in turn implies that $c_n^{1/n} \to 2+2\sqrt{5}$.

Keywords: Pattern avoiding permutation; Algebraic generating function; Context-free language

1 Introduction

Let $p = p_1p_2 \dots p_n$ and $q = q_1q_2 \dots q_k$ be permutations of length $n \geqslant k$. We say p *avoids* q if there are no k indices $i_1 < \cdots < i_k$ so that for all s, t,

$$p_{i_s} < p_{i_t} \qquad \text{if and only if} \qquad q_s < q_t.$$

For example, 25413 avoids 123 since it has no increasing subsequence of length 3.

Interest in sets of permutations that avoid a small set of "patterns" arose naturally in the study of stack-sorting (or equivalently stack-generating) algorithms. Knuth showed that a permutation p can be generated by passing the ordered sequence $12 \dots |p|$ through an infinite stack if and only if p avoids 312, and that permutations of length n avoiding 312 are counted by the Catalan numbers [15].

If $\mathfrak{q}$ is a list of permutations, let $Av_n(\mathfrak{q})$ be the set of permutations of length n that avoid q for each $q \in \mathfrak{q}$. We call $Av(\mathfrak{q}) = \bigcup_{n=0}^{\infty} Av_n(\mathfrak{q})$ a *pattern-avoidance class.* A *basis* for a pattern avoidance class $Av(\mathfrak{q})$ is a set $\mathfrak{p}$ of pairwise avoiding permutations so that $Av(\mathfrak{p}) = Av(\mathfrak{q})$. A class is *finitely based* if it is equal to $Av(\mathfrak{p})$ for $\mathfrak{p}$ finite.

The list of pattern-avoidance classes for which a generating function for the sequence counting $Av_n(\mathfrak{q})$ has been computed, or shown to be rational, algebraic or non-algebraic, is limited. Classes avoiding a single pattern of length 3 are enumerated by the Catalan numbers [15, 17] and so have an algebraic generating function. For length four, $Av(\{1342\})$ has an algebraic generating function [8], $Av(\{1234\})$ has a generating function that is D-finite but not algebraic [13], and a closed form generating function for $Av(\{1324\})$ has not be found [2, 9]. It is known that for any pattern p of length four, $Av(\{p\})$ is in bijection with one of these three classes. For single patterns of length greater than four, and classes avoiding two or more patterns, various isolated results are known [4, 18].

In this article we consider the class of permutations generated by passing an ordered sequence through a stack of depth 2 and infinite stack is series, which was shown by the first author to have a finite basis consisting of 20 patterns [11]. The more general case of two infinite stacks in series has not been enumerated. Pierrot and Rossin recently proved a polynomial time algorithm to decide if a permutation can be sorted by two stacks in series [16]. A special case where both stacks are required to remain ordered was solved by Atkinson, Murphy and Ruškuc [7]. The number of permutations sortable by 2 stacks in *parallel* was recently solved by Albert and Bousquet-Mélou [3].

Several authors have considered the language-theoretic complexity of pattern avoidance classes — see for example [1, 5, 6, 10]. Atkinson, Livesey, and Tulley [6] showed that the set of permutations generated by passing an ordered sequence through a finite *token-passing network* is in bijection with a regular language. Initially we applied this technique to the finite network consisting of a stack of depth 2 followed by a stack of depth k in series, constructing a sequence of languages and corresponding rational generating functions for small values of k. As k increased, the rational generating functions appeared to converge to the algebraic function given in Theorem 11 below. However, this method does not constitute a proof. To prove the result we instead follow another path — we establish a bijection between permutations generated and an unambiguous context-free language. The generating function is then guaranteed to be algebraic by a well known theorem of Chomsky and Schützenberger.

The main work in this article is to establish the bijection with the context-free language. It has been suggested that the method employed to transform the relatively simple pushdown-automaton description of the language to the quartic generating function should be much easier than the method we detail here. We would welcome any insights into this — in our approach we merely apply the standard theory, and give the details for an interested reader.

2 Establishing a bijection

Let $\mathcal{P}$ be the set of permutations that can be generated by a stack of depth 2 and infinite stack in series, and fix ρ, λ, μ as the stack moves indicated in Figure 1.

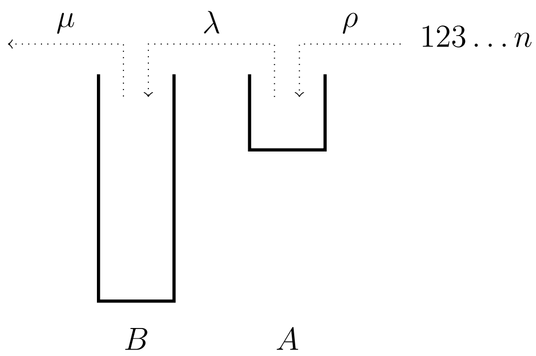

Figure 1: Token passing moves ρ, λ and μ for two stacks in series.

Definition 1 ($\boldsymbol{D_{a,b}(u)}$). If u is a word over an alphabet that includes the letters a and b, define $D_{a,b}(u)$ to be the number of a letters minus the number of b letters contained in u.

Definition 2 ($\boldsymbol{\mathcal{L}_{k,\infty}}$). Let $k \in \mathbb{N}$. The language $\mathcal{L}_{k,\infty}$ is the set of words $w \in \{\rho, \lambda, \mu\}^*$ satisfying

1. $D_{\rho,\lambda}(u) \in [0, k]$ and $D_{\lambda,\mu}(u) \in [0, \infty)$ for all prefixes, u, of w,
2. $D_{\rho,\lambda}(w) = D_{\lambda,\mu}(w) = 0$.

Lemma 3. *A word $w \in \{\rho, \lambda, \mu\}^*$ encodes a permutation in $\mathcal{P}$ if and only if $w \in \mathcal{L}_{2,\infty}$. Moreover, a word of length $3n$ in $\mathcal{L}_{2,\infty}$ encodes a permutation of length n.*

Proof. The first claim is clear from the definition. If $w \in \mathcal{L}_{2,\infty}$ has n ρ letters, then $D_{\rho,\lambda}(w) = 0$ implies w has n λ letters, and $D_{\lambda,\mu}(w) = 0$ then implies w has n μ letters, so the length of w is $3n$, and the number of tokens moves through the stacks is n. $\square$

The language $\mathcal{L}_{2,\infty}$ consists of all possible ways to pass tokens through the system of stacks as in Figure 1. We wish to find a sublanguage that is in bijection with $\mathcal{P}$. From the set of all words in $\mathcal{L}_{2,\infty}$ that generate the same permutation, we will try to choose the string that outputs tokens as soon as possible, that is, has more μ letters closer to the front. The next definition will help to formalise this.

Definition 4 (**$\boldsymbol{\mu}$-ordering**)**.** Define an ordering, $\prec_\mu$, on words in $\{\rho,\lambda,\mu\}^*$ as follows. Let $\theta : \{\rho,\lambda,\mu\}^* \to \{\nu,\mu\}^*$ be a monoid homomorphism defined by $\theta(\mu) = \mu$ and $\theta(\rho) = \theta(\lambda) = \nu$. If $u \neq v$ as strings then $u \prec_\mu v$ if $|u| = |v|$ and $\theta(u)$ precedes $\theta(v)$ in lexographic ordering on $\{\mu,\nu\}^*$ where $\mu < \nu$.

For example, if $u = \rho\lambda\mu\rho\lambda\mu$ and $v = \rho\lambda\rho\mu\lambda\mu$ then $u \prec_\mu v$. Note that both words generate the permutation 12, and u is obtained from v by replacing the subword $\rho\mu$ by $\mu\rho$, which has no affect on the permutation being produced. More generally we have the following.

Lemma 5. *Let $w \in \mathcal{L}_{2,\infty}$.*

1. *If $w = w_0\rho\mu w_1$ then $w' = w_0\mu\rho w_1$ generates the same permutation as w, and $w' \prec_\mu w$.*

2. *If $w = w_0\rho\lambda w_1\lambda\mu w_2$ with $D_{\rho,\lambda}(w_0) = 1$ and $w_1 \in \mathcal{L}_{1,\infty}$, then $w' = w_0\lambda\rho w_1\mu\lambda w_2$ generates the same permutation as w, and $w' \prec_\mu w$.*

3. *If $w = w_0\lambda\rho w_1\lambda\mu w_2$ with $D_{\rho,\lambda}(w_0) = 1$ and $w_1 \in \mathcal{L}_{1,\infty}$, then $w' = w_0\rho\lambda w_1\mu\lambda w_2$ generates the same permutation as w, and $w' \prec_\mu w$.*

Proof. In each case it is clear that $w' \prec_\mu w$. We must show that in each case the two strings generate the same permutation. For case (1) this is clear since ρ and μ do not interact.

For case (2), since $D_{\rho,\lambda}(w_0) = 1$, there must be one token (say a) left in the first stack after reading w_0, and since the next letter to be read is ρ, there must be one token (say b) ready to enter the first stack. See Figure 2.

After reading $\rho\lambda$, b moves to the top of stack B and a stays in stack A. Reading w_1 leaves a and b in place and outputs some permutation of input tokens. Finally $\lambda\mu$ outputs a, leaving b on the top of stack B and stack A empty.

Starting from the initial configuration in Figure 2, the prefix $w_0\lambda\rho w_1\mu\lambda$ of w' moves a to the top of stack B and places b in stack A. The permutation generated by w_1 is then passed across as before, then a is output, and finally b is moved to stack B, leaving the stacks in the same configuration as the prefix $w_0\rho\lambda w_1\lambda\mu$ of w.

A similar argument applies for Case (3) and is left to the reader. □

Definition 6 (**$\mathcal{L}$**)**.** The language $\mathcal{L}$ is the set of words $w \in \mathcal{L}_{2,\infty}$ that do not

1. contain a subword $\rho\mu$,

2. have a prefix $w_0\rho\lambda w_1\lambda\mu$ with $w_1 \in \mathcal{L}_{1,\infty}$ and $D_{\rho,\lambda}(w_0) = 1$,

3. have a prefix $w_0\lambda\rho w_1\lambda\mu$ with $w_1 \in \mathcal{L}_{1,\infty}$ and $D_{\rho,\lambda}(w_0) = 1$.

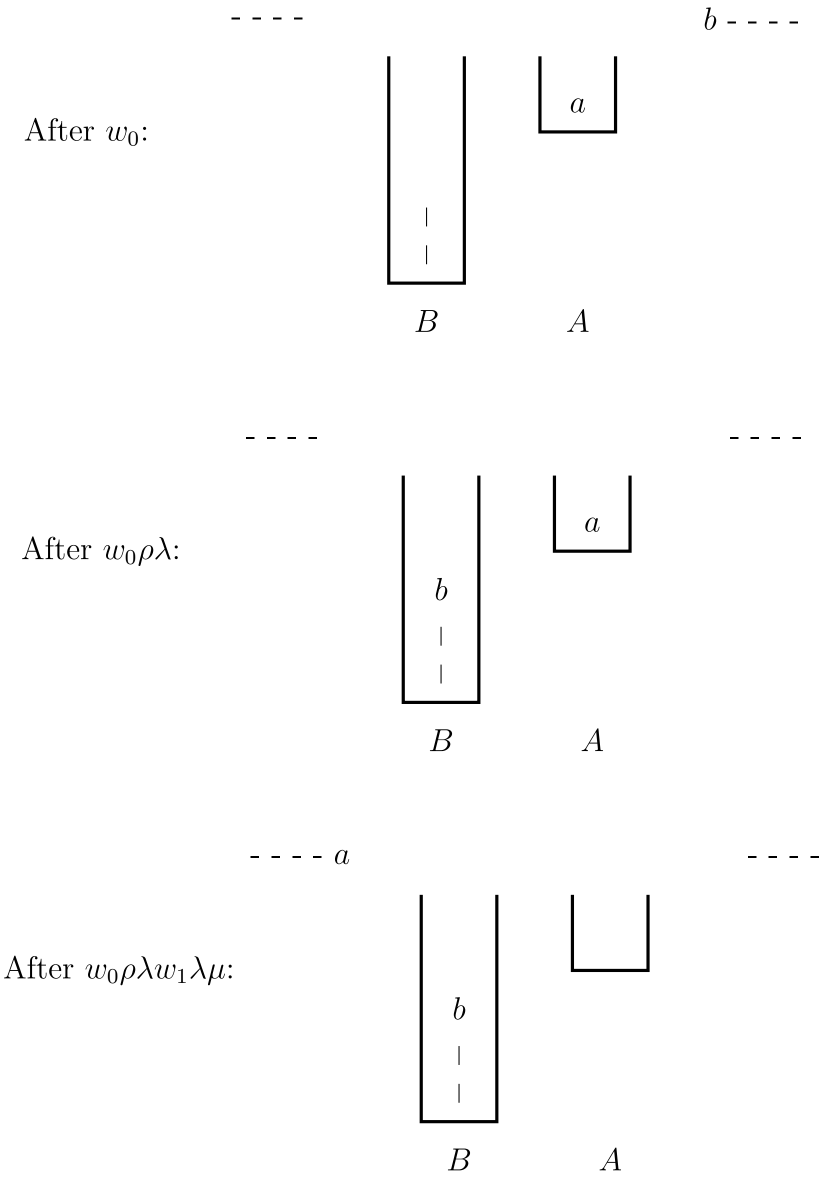

Figure 2: Stack configurations in the proof of Lemma 5.

Lemma 7. *Let $w \in \mathcal{L}_{2,\infty}$. If either*

1. $w = w_0\rho\lambda w_1\lambda w_2\mu w_3$ *with* $D_{\rho,\lambda}(w_0) = 1, w_1 \in \mathcal{L}_{1,\infty}$*, and* $w_2 \in \mathcal{L}_{2,\infty}$ *generates a permutation that avoids* 312*, or*

2. $w = w_0\lambda\rho w_1\lambda w_2\mu w_3$ *with* $D_{\rho,\lambda}(w_0) = 1, w_1 \in \mathcal{L}_{1,\infty}$*, and* $w_2 \in \mathcal{L}_{2,\infty}$ *generates a permutation that avoids* 312*,*

then $w \notin \mathcal{L}$*.*

Proof. Suppose for contradiction that $w \in \mathcal{L}$, $w = w_0vw_1\lambda w_2\mu w_3$ with $v \in \{\rho\lambda, \lambda\rho\}$, $D_{\rho,\lambda}(w_0) = 1, w_1 \in \mathcal{L}_{1,\infty}, w_2$ generates a permutation that avoids 312, and moreover that w_0 is the longest prefix of w with this property. That is, if $w = u_0vu_1\lambda u_2\mu u_3$ with $v \in \{\rho\lambda, \lambda\rho\}, D_{\rho,\lambda}(u_0) = 1, u_1 \in \mathcal{L}_{1,\infty}$ and u_2 generates a permutation that avoids 312, then $|u_0| \leqslant |w_0|$.

Since $D_{\rho,\lambda}(w_0vw_1) = 1$ and λ moves a token from stack A to stack B, after reading $w_0vw_1\lambda$ we have no tokens in stack A, and some token, say a, in stack B. See Figure 3.

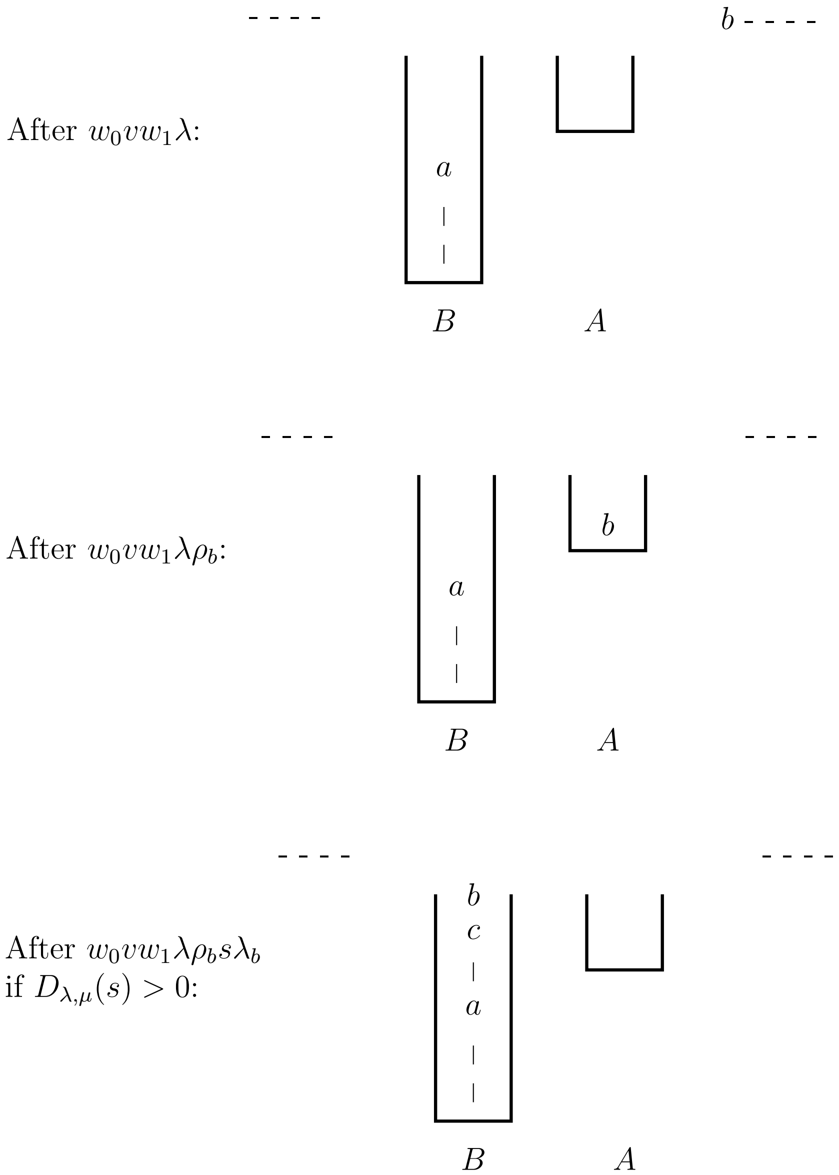

Figure 3: Stack configurations in the proof of Lemma 7.

Since $w \in \mathcal{L}$, w_2 cannot be empty, and since w_2 is a subword of $w \in \mathcal{L}$ we have

$w_2 \in \mathcal{L}$. So w_2 moves some sequence of tokens completely through the stacks, leaving a in place. The first letter of w_2 must be ρ, which moves some token, say b, onto stack A. Let ρ_b, λ_b, μ_b be the letters in w_2 that correspond to moving b through the stacks. Then w_2 has prefix $\rho_b s \lambda_b t \mu_b$ where s, t are subwords.

Since stack A contains b while s is read, if ρ occurs in s it must be immediately followed by λ, so $D_{\rho,\lambda}(u) \in [0,1]$ for all prefixes u of s, and $D_{\rho,\lambda}(s) = 0$. Further, if $D_{\lambda,\mu}(u) < 0$ for any prefix u of s, then a would be output. Either $D_{\lambda,\mu}(s) = 0$ (and $s \in \mathcal{L}$) or $D_{\lambda,\mu}(s) > 0$.

If $s \in \mathcal{L}_{1,\infty}$ then $t \in \mathcal{L}_{2,\infty}$ and generates a permutation avoiding 312 since it is a subword of w_2. In this case w has prefix $w = w_0 v w_1 \lambda \rho_b s \lambda_b t \mu_b$ with $D_{\lambda,\mu}(w_0 v w_1) = 1$ and t generating a permutation avoiding 312, which contradicts the choice of w_0 as the longest such prefix.

Therefore we must have $D_{\lambda,\mu}(s) > 0$. In this case, after reading s at least one token, say c, remains on top of a in stack B when b is moved into it. After reading λ_b, the stack configuration is as in the third diagram shown in Figure 3.

Note that $a < b < c$ since they are input in this order. Let ε denote the empty word. If $t \neq \varepsilon$ then it must contain at least one μ (it cannot leave a token covering b, and cannot just be ρ or $\rho\rho$) so it moves a token $d > c$ to the output. This means w_2 generates the subpermutation dbc which is order equivalent to 312, contradicting our assumption. Thus $t = \varepsilon$ and w_2 has prefix $\rho_b s \lambda_b \mu_b$, with $s \in \{\rho\lambda, \mu\}^*$. Either s ends with $\rho\lambda$, or $s = u\rho\lambda s'$ where $D_{\lambda,\mu}(u) = D_{\lambda,\mu}(s)$ since $D_{\lambda,\mu}$ starts at zero and increases to this value. Thus $s' \in \mathcal{L}$, and $w = w_0 v w_1 \lambda \rho_b u \rho \lambda s' \lambda_b \mu_b$ with $D_{\rho,\lambda}(w_0 v w_1 \lambda \rho_b u) = 1$, which contradicts $w \in \mathcal{L}$. $\square$

Theorem 8. *There is a bijection between permutations in $\mathcal{P}$ of length n and words in $\mathcal{L}$ of length $3n$.*

Proof. Consider the map that sends a word of length $3n$ in $\mathcal{L} \subseteq \mathcal{L}_{2,\infty}$ to the permutation of length n it generates. If $\sigma \in \mathcal{P}$ then there is some word $w \in \mathcal{L}_{2,\infty}$ that generates it by Lemma 3. If $w \notin \mathcal{L}$, then w must either contain $\rho\mu$, or have prefix $w_0\rho\lambda w_1\lambda\mu$ or $w_0\lambda\rho w_1\lambda\mu$ with $D_{\rho,\lambda}(w_0) = 1$ and $w_1 \in \mathcal{L}_{1,\infty}$. We rewrite w as follows.

While w contains $\rho\mu$ or has prefix $w_0\rho\lambda w_1\lambda\mu$ or $w_0\lambda\rho w_1\lambda\mu$:

1. Replace $\rho\mu$ with $\mu\rho$
2. Replace $w_0\rho\lambda w_1\lambda\mu$ with $w_0\lambda\rho w_1\mu\lambda$
3. Replace $w_0\lambda\rho w_1\lambda\mu$ with $w_0\rho\lambda w_1\mu\lambda$

Each iteration replaces the current word by a word which generates the same permutation and is shorter in the μ-ordering by Lemma 5, so the procedure must terminate (there are finitely many words less than w in the μ-ordering). It follows that the map is surjective. We complete the proof by showing it is injective.

Suppose we have two words $u, v \in \mathcal{L}$ that generate the same permutation, and that $u \neq v$ as strings. Write

$$u = u_1 u_2 \dots u_n \text{ and } v = v_1 v_2 \dots v_n$$

where $u_i, v_i \in \{\rho, \lambda, \mu\}$.

Since $u, v \in \mathcal{L}$ we have $u_1 = v_1 = \rho$. Let $k \in [2, n]$ be such that $u_i = v_i$ for $i < k$ and $u_k \neq v_k$. Let $z = u_1 \dots u_{k-1} = v_1 \dots v_{k-1}$, so

$$u = zu_k \dots u_n \text{ and } v = zv_k \dots v_n.$$

First consider the case that one of u_k, v_k is μ. Without loss of generality assume $u = z\mu u_{k+1} \dots u_n$. Then z must leave some token, say a, at the top of stack B, and $u_k = \mu$ outputs this token.

If $v_k = \lambda$, then a will be covered and v will not be able to generate the same permutation. So we must have $v_k = \rho$. Then $v_{k+1} \neq \mu$. If $v_{k+1} = \lambda$ then a is covered. So $v_{k+1} = \rho$. Then $v_{k+2} \neq \mu$, if $v_{k+2} = \lambda$ then a is covered, and $v_{k+2} \neq \rho$ since stack A contains two tokens. So we have a contradiction, and it follows that neither u_k, v_k can be μ.

Without loss of generality assume $u_k = \rho$ and $v_k = \lambda$. Then z must leave at least one token in stack A to be followed by λ, and at most one token in stack A to be followed by ρ. Let a be the token in A, and b the token moved from the input by $u_k = \rho$. See Figure 4. Note that we have $D_{\rho,\lambda}(z) = 1$.

In u, $z\rho$ must be followed by λ since stack A is full after the ρ and ρ cannot be followed by a μ. So u has prefix $z\rho\lambda$ and we have the configuration shown in the second diagram in Figure 4.

In v, $z\lambda$ can be followed by either μ or ρ but not λ since stack A is empty after $v_k = \lambda$. Suppose $v_{k+1} = \mu$. Then after reading $z\lambda\mu$ we have the configuration shown in the third diagram in Figure 4. Since u and v are assumed to produce the same permutation, the next μ letter appearing in u after the prefix $z\rho\lambda$ must move a to the output. Let λ_a, μ_a be the letters in u that move the token a. Then u has prefix $z\rho\lambda\kappa_1\lambda_a\kappa_2\mu_a$ where $\kappa_1, \kappa_2 \in \{\rho, \lambda\}^*$. The subword κ_2 cannot move tokens to cover a in stack B, so cannot contain any λ letters, and cannot contain any ρ letters since it is followed by μ, so it must be empty. The subword κ_1 must be of the form $(\rho\lambda)^i$ for $i \geqslant 0$, since it cannot move a. So u has prefix $z(\rho\lambda)^i\rho\lambda\lambda_a\mu_a$ with $D_{\rho,\lambda}(z(\rho\lambda)^i) = 1$, so $u \notin \mathcal{L}$.

It follows that $v_{k+1} = \rho$, so we have

$$u = z\rho\lambda u_{k+2} \dots u_n, v = z\lambda\rho v_{k+2} \dots v_n.$$

The two configurations of the stacks after reading the length $k+1$ prefixes of u and v respectively are shown in Figure 5.

We now consider two possibilities: either a precedes b in the permutation generated by u and v, or b precedes a.

Case 1: *a* precedes *b*

Mark the letters λ, μ in u and v that correspond to moving the token a by appending the subscript a. So u has prefix $z\rho\lambda w_1\lambda_a w_2\mu_a$ and v has prefix $z\lambda_a\rho w\mu_a$ where $w, w_1, w_2 \in \{\rho, \lambda, \mu\}^*$.

First consider the word v. Since w cannot end with ρ, and b must remain in stack A until a is output, and w cannot leave any tokens covering a in stack B, and we have

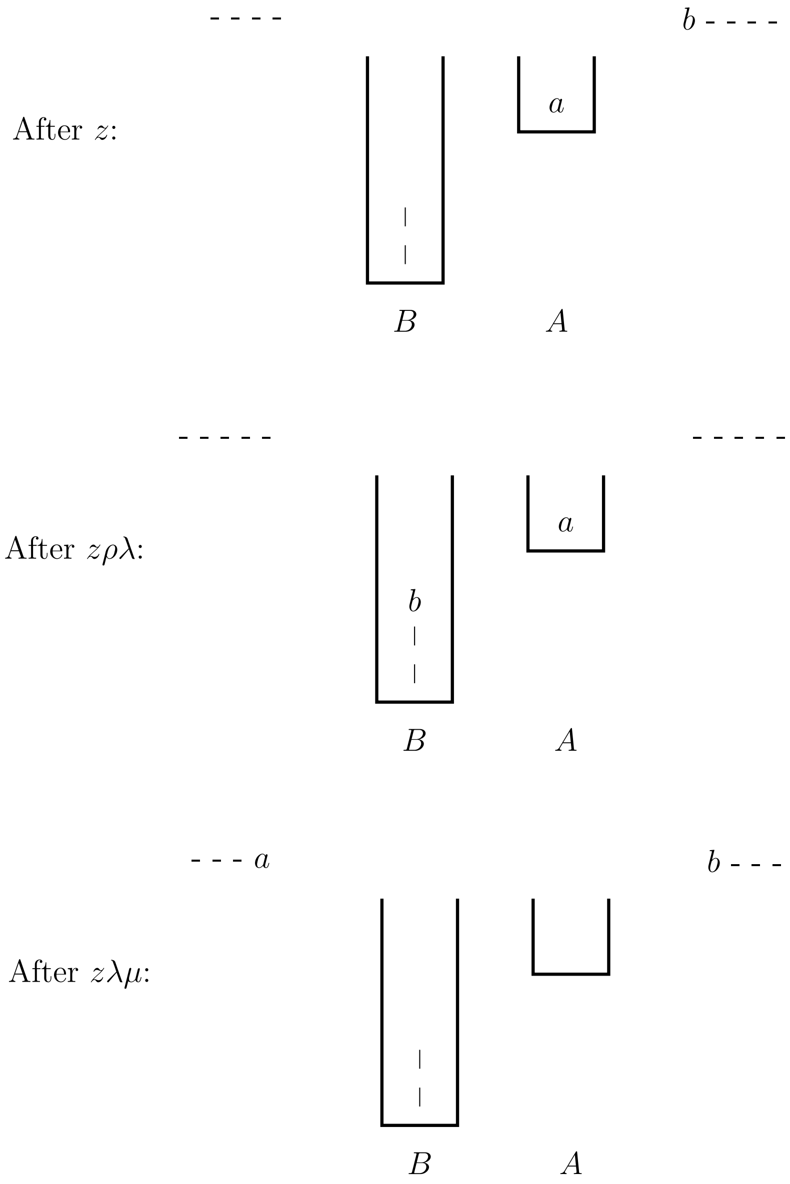

Figure 4: Stack configurations in Theorem 8 where $u_k = \rho$ and $v_k = \lambda$.

$w \in \mathcal{L}_{1,\infty}$. If w is empty then v contains $\rho\mu_a$ which means $v \notin \mathcal{L}$. Thus w is nonempty, so moves some tokens, say $t_1, \ldots, t_s$, from the input to the output.

Since u generates the same permutation as v, it must also move the tokens $t_1, \ldots, t_s$ through the stacks and output them before a is output. The subword w_1 cannot leave any tokens covering a in stack A, so $w_1 \in \{\rho\lambda, \mu\}^*$. If w_1 leaves some token t_i in stack B, then λ_a will cover it by a, which means t_i will appear after a in the output, so u does not generate the same permutation as v. Thus after w_1 the stacks are the same as before w_1, so $w_1 \in \mathcal{L}_{1,\infty}$. Let $t_1, \ldots t_r$ with $r \leqslant s$ be the tokens moved to the output by w_1. The

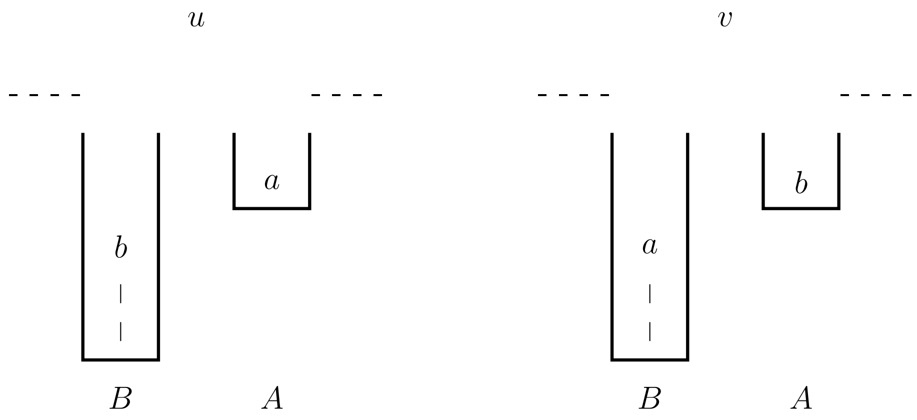

Figure 5: Stack configurations after $z\rho\lambda$ and $z\lambda\rho$ in Theorem 8.

situtation after reading w_1 is shown in Figure 6.

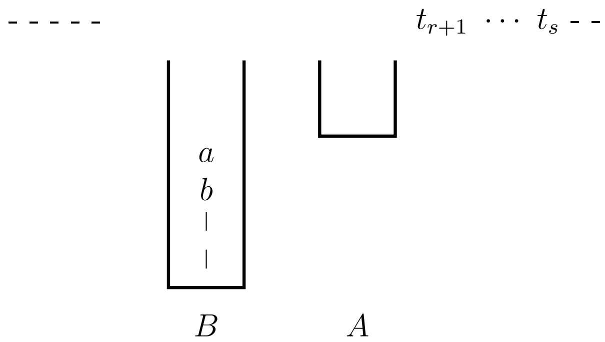

Figure 6: Stack configuration after $z\rho\lambda w_1\lambda_a$ in Case 1 in Theorem 8.

If w_2 is empty then u has prefix $z\rho\lambda w_1\lambda_a\mu_a$ with $w_1 \in \mathcal{L}_{1,\infty}$ which is forbidden, so w_2 must move some tokens. Also w_2 cannot leave any tokens in stack B. Either w_2 leaves some tokens in stack A, or not.

If w_2 leaves a token in stack A, this token cannot be one of $t_{r+1}, \ldots, t_s$ or else u would generate a different permutation to v. Therefore this token is moved into stack A after t_r by a letter ρ. This letter cannot be followed by μ, and since it remains in stack A it is not followed by λ. So this letter is either the last letter of w_2, or is followed by another ρ, which must also remain in stack A. Thus w_2 ends with ρ, but this is a contradiction since w_2 is followed by μ_a.

Thus w_2 does not leave any tokens in stacks A or B, so moves $t_{r+1}, \ldots, t_s$ from the input to the output, so $w_2 \in \mathcal{L}_{2,\infty}$. Note that w_1w_2 produces the same permutation of $t_1, \ldots, t_s$ as w does, and $w \in \mathcal{L}_{1,\infty}$ so generates a 312-avoiding permutation of $t_1, \ldots, t_s$. The subword w_1 permutes the first r tokens, and so w_2 must produce a permutation of $t_{r+1}, \ldots, t_s$ that avoids 312. In this case u has prefix $z\rho\lambda w_1\lambda_a w_2\mu_a$ where $D_{\rho,\lambda}(z\rho\lambda) = 1$,

$w_1 \in \mathcal{L}_{1,\infty}$ and w_2 generates a 312-avoider, so by Lemma 7 u must also contain a prefix that is not allowed if $u \in \mathcal{L}$. This is a contradiction, so this case does not apply.

Case 2: *b* precedes *a*

We return to the situation shown in Figure 5 with $u = z\rho\lambda u_{k+2} \dots u_n$ and $v = z\lambda\rho v_{k+2} \dots v_n$. Mark the letters ρ, λ, μ in u and v that correspond to moving the token b by appending a subscript b. Then u has prefix $z\rho_b\lambda_b w\mu_b$ and v has prefix $z\lambda\rho_b w_1\lambda_b w_2\mu_b$ where $w, w_1, w_2 \in \{\rho, \lambda, \mu\}^*$.

First consider the word u. Since w is followed immediately by μ_a, w cannot end with a ρ, and since a must remain in stack A until b is output and w cannot leave any tokens covering b in stack B, we have $w \in \mathcal{L}_{1,\infty}$.

If w is empty then u has prefix $z\rho_b\lambda_b\mu_b$. For v to produce the same permutation the next token it must output from the configuration shown in Figure 5 is b, so $w_1 \in \{\rho\lambda\}^*$ and $w_2 \in \{\rho\}^*$. Since w_2 is followed by μ_b then w_2 must be empty. Then v has prefix $z\lambda\rho_b(\rho\lambda)^i\lambda_b\mu_b$ for some $i \geqslant 0$. If $i = 0$ then v has prefix $z\lambda\rho_b\lambda_b\mu_b$ with $D_{\rho,\lambda}(z) = 1$ which violates the third condition of Definition 6, and if $i > 0$ then v has prefix $(z\lambda\rho_b(\rho\lambda)^{i-1})\rho\lambda\lambda_b\mu_b$ with $D_{\rho,\lambda}(z\lambda\rho_b(\rho\lambda)^{i-1}) = 1$ which violates the second condition of Definition 6, so $v \notin \mathcal{L}$.

Thus w is nonempty, so moves some tokens, say $t_1, \dots, t_s$, from the input to the output. Since v generates the same permutation as u, it must also move the tokens $t_1, \dots, t_s$ through the stacks and output them before b is output. The subword w_1 cannot leave any tokens covering b in stack A, so $w_1 \in \{\rho\lambda, \mu\}^*$.

If w_1 leaves some token t_i in stack B, then λ_b will cover t_i with b so t_i will appear after b in the permutation generated by v, which is different to the permutation generated by u. Thus after w_1 the stacks are the same as before w_1, so $w_1 \in \mathcal{L}_{1,\infty}$. Let $t_1, \dots t_r$ with $r \leqslant s$ be the tokens moved to the output by w_1. The situation is shown in Figure 7.

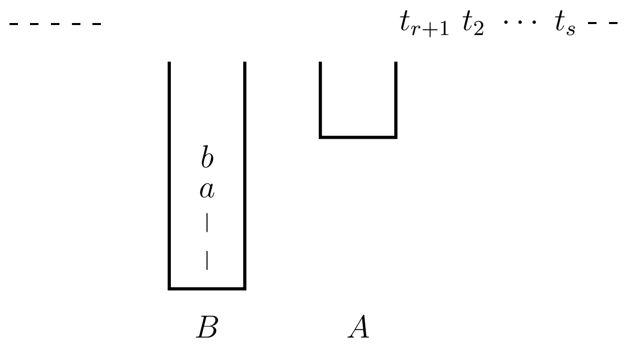

Figure 7: Stack configuration after $z\lambda\rho_b w_1\lambda_b$ in Case 2 in Theorem 8.

If w_2 is empty then v has prefix $z\lambda\rho w_1\lambda_b\mu_b$ with $w_1 \in \mathcal{L}_{1,\infty}$ which is forbidden, so w_2 must move some tokens. The subword w_2 cannot leave any tokens in stack B. Either w_2 leaves some tokens in stack A, or not.

If w_2 leaves a token in stack A, this token cannot be one of $t_{r+1}, \dots, t_s$ or else v would generate a different permutation to u. Therefore this token is moved into stack A after

t_r by a letter ρ. This letter cannot be followed by μ, and since it remains in stack A it is not followed by λ. So this letter is either the last letter of w_2, or is followed by another ρ, which must also remain in stack A. Thus w_2 ends with ρ, but this is a contradiction since w_2 is followed by μ_b.

Thus w_2 does not leave any tokens in stacks A or B, so moves $t_{r+1}, \ldots, t_s$ from the input to the output, and $w_2 \in \mathcal{L}_{2,\infty}$. Note that $w_1 w_2$ produces the same permutation of $t_1, \ldots, t_s$ as w does, and $w \in \mathcal{L}_{1,\infty}$ so generates a 312-avoiding permutation of $t_1, \ldots, t_s$. The subword w_1 permutes the first r tokens, and so w_2 must produce a permutation of $t_{r+1}, \ldots, t_s$ that avoids 312. In this case v has prefix $z\lambda\rho_b w_1 \lambda_b w_2 \mu_b$ where $D_{\rho,\lambda}(z\lambda\rho_b) = 1$, $w_1 \in \mathcal{L}_{1,\infty}$ and w_2 generates a 312-avoider, so by Lemma 7 v must also contain a prefix that is not allowed if $v \in \mathcal{L}$. This is a contradiction, so we cannot have two such words u and v. $\square$

2.1 A related class of permutations

A natural question to ask is whether switching the order of the stacks makes any difference to the problem. Let $\mathcal{Q}$ be the set of permutations that can be generated by passing an ordered sequence through an infinite stack followed by a depth 2 stack in series. Each word $w \in \mathcal{L}_{2,\infty}$ encodes a permutation in $\mathcal{Q}$ as follows: reading w from right to left, for each μ move a token from the input to the infinite stack, for each λ move a token from the infinite stack to the depth 2 stack, and for each ρ move a token from the depth 2 stack to the output. It follows that $\mathcal{P}$ and $\mathcal{Q}$ are in bijection.

3 Constructing a pushdown automaton

In this section we construct a deterministic pushdown automaton accepting on empty stack, which accepts the language

$$\mathcal{L}\$ = \{w\$ \mid w \in \mathcal{L}\}.$$

A *pushdown automaton accepting on empty stack M* is the following:

1. Q a finite set of *states*,
2. Σ a finite *input alphabet*,
3. Γ a finite *stack alphabet*,
4. $q_0 \in Q$ the *start state*,
5. $0 \in \Gamma$ a special stack symbol,
6. a map δ from $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to finite subsets of $Q \times (\Gamma^*)$, called the *transition function*,

which runs as follows. Before reading input, the stack contains a single 0. Input strings are accepted as soon as the stack becomes empty. A *configuration* of M is a pair (q, ω) where q is the current state and $\omega \in \Gamma^*$ is a string of stack symbols representing the contents of the stack (the first letter of ω is the top of the stack). The notation $\delta(q_i, a, k) = \{(q_{j_1}, \gamma_1), \ldots, (q_{j_s}, \gamma_s)\}$ means that if M has the configuration $(q_i, k\omega)$ and $a \in \Sigma \cup \{\varepsilon\}$ is the next input letter to be read, then M can move to the configuration $(q_{j_l}, \gamma_l\omega)$ for some $1 \leqslant l \leqslant s$, removing the token k from the top of the stack and replacing it by γ_l.

See [14] for more details.

A pushdown automaton is *deterministic* if for each state q and stack symbol i

1. if $|\delta(q, \varepsilon, i)| = 1$ then $|\delta(q, a, i)| = 0$ for all $a \in \Sigma$,
2. for each $a \in \Sigma \cup \{\varepsilon\}$ the set $\delta(q, a, i)$ has size at most one.

Note that a determistic pushdown automaton accepting on empty stack cannot accept the empty string (unless this is the only string it accepts) since there would have to be a transition $\delta(q_0, \varepsilon, 0)$ as well as a transition $\delta(q_0, a, 0)$ for some letter a.

Let M be the pushdown automaton shown in Figure 8, which accepts on empty stack. M uses its stack to keep track of $D_{\lambda,\mu}$ as it reads its input, and its states to keep track of $D_{\rho,\lambda}$. It uses the stack symbol 2 as a device to flag when the input has the potential to have a prefix of the form $w_0\rho\lambda$ or $w_0\lambda\rho$ with $D_{\rho,\lambda}(w_0) = 1$. Paths $\rho\mu$ are forbidden. We will prove that the language of this automaton is precisely the language $\mathcal{L}\$$.

Here is the formal description of M. Note that states q_3, q_6, q_7 are reached only when 1 is on top of the stack, and q_5, q_8 are reached when either 1 or 2 are on top of the stack, so we have omitted transitions from configurations that are not possible.

1. states $Q = \{q_0, \ldots, q_8\}$,
2. input alphabet $\Sigma = \{\rho, \lambda, \mu, \$\}$,
3. stack alphabet $\Gamma = \{0, 1, 2\}$,
4. start state q_0,
5. transition function δ defined as follows.

$\delta(q_0, \$, 0) = (q_0, \varepsilon)$	$\delta(q_1, \rho, 1) = (q_2, 1)$	$\delta(q_1, \lambda, 2) = (q_3, 12)$
$\delta(q_0, \mu, 1) = (q_0, \varepsilon)$	$\delta(q_1, \rho, 2) = (q_2, 2)$	$\delta(q_2, \lambda, 0) = (q_5, 10)$
$\delta(q_0, \mu, 2) = (q_0, \varepsilon)$	$\delta(q_3, \rho, 1) = (q_6, 1)$	$\delta(q_2, \lambda, 1) = (q_5, 11)$
$\delta(q_3, \mu, 1) = (q_0, \varepsilon)$	$\delta(q_4, \rho, 0) = (q_2, 0)$	$\delta(q_2, \lambda, 2) = (q_5, 12)$
$\delta(q_4, \mu, 1) = (q_4, \varepsilon)$	$\delta(q_4, \rho, 1) = (q_2, 1)$	$\delta(q_4, \lambda, 0) = (q_3, 10)$
$\delta(q_4, \mu, 2) = (q_4, \varepsilon)$	$\delta(q_4, \rho, 2) = (q_2, 2)$	$\delta(q_4, \lambda, 1) = (q_3, 11)$
$\delta(q_5, \mu, 1) = (q_4, \varepsilon)$	$\delta(q_5, \rho, 1) = (q_8, 1)$	$\delta(q_4, \lambda, 2) = (q_3, 12)$
$\delta(q_5, \mu, 2) = (q_5, \varepsilon)$	$\delta(q_5, \rho, 2) = (q_8, 2)$	$\delta(q_5, \lambda, 1) = (q_7, 11)$
$\delta(q_0, \rho, 0) = (q_1, 0)$	$\delta(q_6, \rho, 1) = (q_8, 1)$	$\delta(q_5, \lambda, 2) = (q_7, 12)$
$\delta(q_0, \rho, 1) = (q_1, 1)$	$\delta(q_7, \rho, 1) = (q_6, 1)$	$\delta(q_6, \lambda, 1) = (q_7, 11)$
$\delta(q_0, \rho, 2) = (q_1, 2)$	$\delta(q_1, \lambda, 0) = (q_3, 10)$	$\delta(q_8, \lambda, 1) = (q_5, 21)$
$\delta(q_1, \rho, 0) = (q_2, 0)$	$\delta(q_1, \lambda, 1) = (q_3, 11)$	$\delta(q_8, \lambda, 2) = (q_5, 22)$

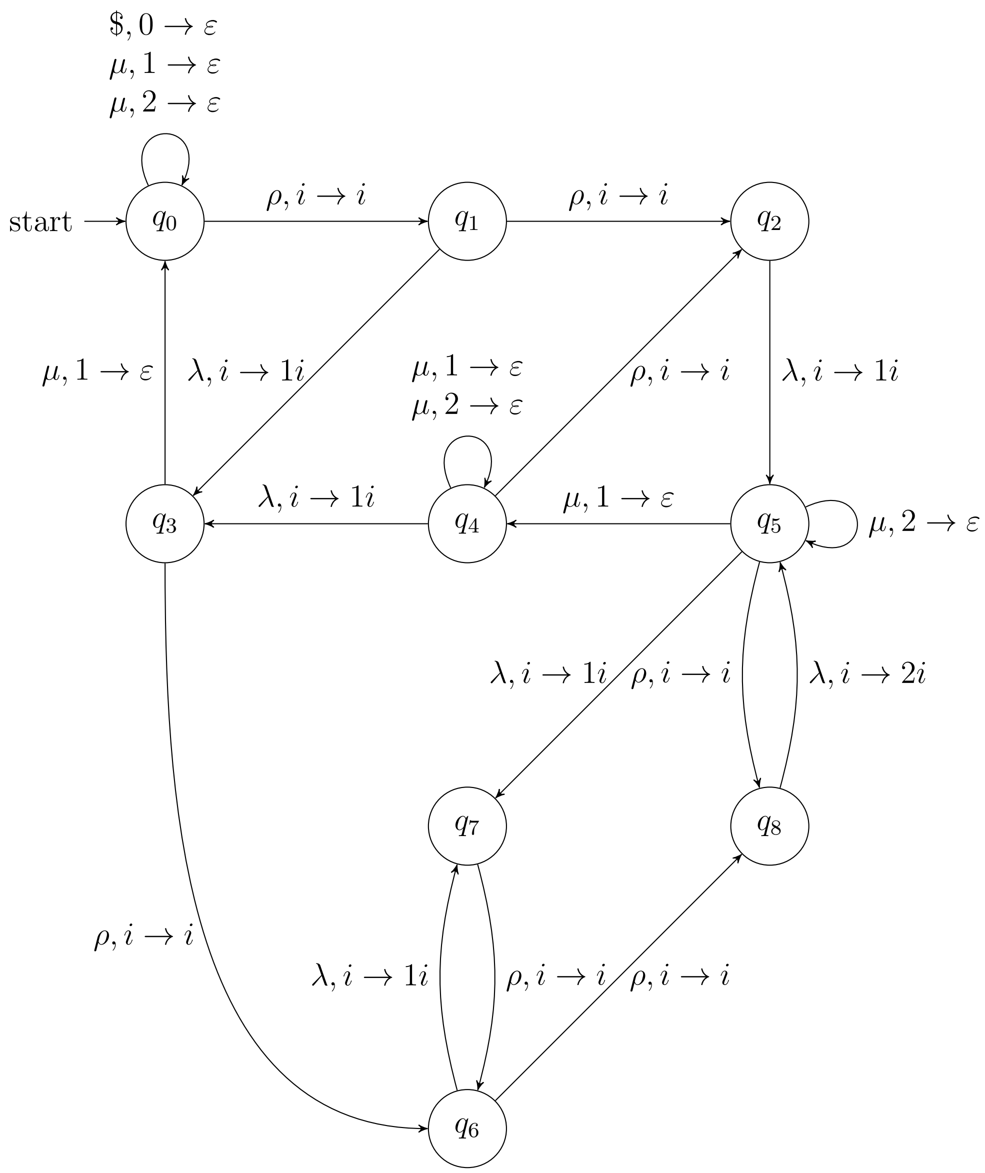

Figure 8: Pushdown automaton M accepting on empty stack, with start configuration $(q_0, 0)$. The symbol $i \in \{0, 1, 2\}$ represents a stack token that is kept in place by a transition.

To prove that M accepts precisely the language $\mathcal{L}\$$, we first show that M is deterministic. This allows us to identify input words with unique paths in M and simplify our arguments slightly.

Lemma 9. *The pushdown automaton M is deterministic.*

Proof. The claim is easily verified by considering the formal description for M. □

Proposition 10. *The pushdown automaton M accepts the language $\mathcal{L}\$ = \{w\$ \mid w \in \mathcal{L}\}$.*

Proof. Since M is deterministic, we identify input words with their corresponding unique path in M.

Let $w \in \{\rho, \lambda, \mu\}^*$. We must show that

1. if w contains $\rho\mu$, then $w\$$ is rejected.
2. if w fails to be in $\mathcal{L}_{2,\infty}$, then $w\$$ is rejected,
3. if w has a bad prefix (conditions (2) and (3) in Definition 6), then $w\$$ rejected.
4. if $w\$$ is rejected, then $w \notin \mathcal{L}$.

The only states that can be reached by a path $u\rho$ for $u \in \{\rho, \lambda, \mu\}^*$ from the start configuration are q_1, q_2, q_6 and q_8 and since none are the source of a μ transition, any word containing $\rho\mu$ will be rejected.

Next, we show that if w is not in $\mathcal{L}_{2,\infty}$, then $w\$$ is rejected by M. Each state represents the endpoint of a path labeling a prefix of an input string accepted by the automaton. One can verify the values of $D_{\rho,\lambda}(u)$ for each path labeled u ending at state q_i given by Table 1.

state	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
$D_{\rho,\lambda}$	0	1	2	0	1	1	1	0	2

Table 1: Value of $D_{\rho,\lambda}$ for any prefix ending at each state

Let $h(u)$ be the height of the second (infinite) stack after reading $u \in \{\rho, \lambda, \mu\}^*$ starting from the start configuration $(q_0, 0)$. Then $h(\varepsilon) = 1$, $h(u\rho) = h(u), h(u\lambda) = h(u) + 1$ and $h(u\mu) = h(u) - 1$ since λ pushes a token to the stack, μ pops a token and ρ keeps the stack unchanged. It follows that $h(u) = D_{\lambda,\mu}(u) + 1$, and since 0 stays on the stack until \$ is read, $h(u) \geqslant 1$ for all prefixes $u \in \{\rho, \lambda, \mu\}^*$, so $D_{\lambda,\mu}(u) \geqslant 0$. If $w\$$ is accepted then the stack must contain only 0 after reading w, so $D_{\lambda,\mu}(w) = 0$.

It follows that if $D_{\rho,\lambda}(u) > 2, D_{\rho,\lambda}(u)$,$D_{\lambda,\mu}(u) < 0$ for some prefix u, or $D_{\lambda,\mu}(w) \neq 0$, then M will reject $w\$$.

Next, suppose $w \in \mathcal{L}_{2,\infty}$ has no $\rho\mu$ substring and a prefix of the form $w_0vw_1\lambda\mu$ where $D_{\rho,\lambda}(w_0) = 1, v \in \{\rho\lambda, \lambda\rho\}$ and $w_1 \in \mathcal{L}_{1,\infty}$. The string w_0 labels a path in the automaton starting at q_0 and ending at state q_1, q_4, q_5 or q_6 by Table 1. From each of these states, reading $v = \rho\lambda$ ends in state q_5, and reading $v = \lambda\rho$ ends in state q_6.

From q_5, the word w_1 labels a path that visits only states q_5 and q_8, since $D_{\lambda,\mu}(z) \geqslant 0$ for all prefixes z of w_1, so the 1 on top of the stack before reading w_1 remains (and is covered by 2s, which are removed by the μ loop at q_5), and ends at q_5 since $D_{\lambda,\mu}(w_1) = 0$. From here reading $\lambda\mu$ is rejected.

From q_6, if $w_1 = \varepsilon$ then $u\lambda\rho\lambda\mu$ is rejected. Otherwise w_1 labels a path from q_6 to q_8 and then moves between q_5 and q_8, and ends at q_5. From here reading $\lambda\mu$ is rejected.

We have now established that if $w \notin \mathcal{L}$ then $w\$$ is rejected by M. To complete the proof we must show that if $w\$$ is rejected, then $w \notin \mathcal{L}$. To show this, assume $w \in \mathcal{L}_{2,\infty}$

with no $\rho\mu$ substring, but $w\$$ is rejected by M. We will prove that w must have a bad prefix.

Let p be the longest prefix of $w\$$ labeling a path that is not rejected by M. Since $w \in \mathcal{L}_{2,\infty}$ we have $D_{\lambda,\mu}(w) = 0$, so if $w = p$, after reading w the stack contains just 0 so $w\$$ will be accepted, a contradiction. Thus p is strictly shorter than w. Let $w = pxw'$ where $x \in \{\rho\lambda, \mu\}$ is the next letter input after reading p.

We now consider the possible states where p can end.

1. Suppose p ends at q_0. Then $D_{\rho,\lambda}(p) = 0$ so $x \neq \lambda$. If the top of stack is 0 then $D_{\lambda,\mu}(p) = 0$ so $x \neq \mu$. Otherwise M cannot reject on reading ρ, μ.

2. Suppose p ends at q_1, so its last letter is ρ, and $D_{\rho,\lambda}(p) = 1$. Then $x \neq \mu$. Otherwise M cannot reject on reading ρ, λ.

3. Suppose p ends at q_2, so its last letter is ρ, and $D_{\rho,\lambda}(p) = 2$. Then $x \neq \mu, \rho$. Otherwise M cannot reject on reading λ.

4. Suppose p ends at q_3, so $D_{\rho,\lambda}(p) = 0$ and the top of stack is 1. Then $x \neq \lambda$. Otherwise M cannot reject on reading $\rho, (\mu, 1 \to \varepsilon)$.

5. Suppose p ends at q_4, so $D_{\rho,\lambda}(p) = 1$. The only way M could reject is if the top of stack is 0 and $x = \mu$, which is not possible since $w \in \mathcal{L}_{2,\infty}$.

6. Suppose p ends at q_5, so $D_{\rho,\lambda}(p) = 1$ and 1 is on top of the stack. Then no letter will cause M to reject.

7. Suppose p ends at q_6, so $D_{\rho,\lambda}(p) = 1$ and p ends with $\lambda\rho$. Then x cannot be μ, and otherwise px is not rejected.

8. Suppose p ends at q_8, so its last letter is ρ, and $D_{\rho,\lambda}(p) = 2$. Then $x \neq \mu, \rho$ and M cannot reject if $x = \lambda$.

These cases show that if p ends at any state except q_7, then M does not reject w on reading the next input letter. We finish the proof by showing that if p ends at q_7, then px is a bad prefix.

Since p ends at q_7, p ends with λ, $D_{\rho,\lambda}(p) = 2$, and $D_{\lambda,\mu}(p) > 0$. If $x = \rho$ then px is not rejected. If $x = \lambda$ then $w \notin \mathcal{L}_{2,\infty}$. So we must have $x = \mu$.

Let $p = p_1\lambda$. If p_1 ends at q_6, then $p_1 = p_2\lambda\rho$, and $px = p_2\lambda\rho\lambda\mu$ where $D_{\rho,\lambda}(p_2) = 1$ and so px is a bad prefix. The machine correctly rejects the string on reading $x = \mu$.

Otherwise p_1 ends at q_5. Either p_1 ends with $\rho\lambda$, or μ. If $p_1 = p_2\rho\lambda$ then $D_{\rho,\lambda}(p_2) = 1$ and $px = p_2\rho\lambda\lambda\mu$ is a bad prefix. Otherwise p_1 ends in μ, and must pop a token 2 from the stack. Let λ_* be the last λ letter in p_1 that pushed a 1 on top of the stack (which must exist, since all paths to q_5 must cross such an edge). Write $p_1 = p_2\lambda_*p_3\mu$.

The letter λ_* labels one of the following edges:

1. from q_2 to q_5,
2. from q_1 to q_3,
3. from q_4 to q_3,
4. from q_5 to q_7,
5. from q_6 to q_7.

In the first case, p_2 ends at q_2 so must have the form $p_2 = u\rho$ with $D_{\rho,\lambda}(u) = 1$. Then $p_3\mu$ labels a path that moves between states q_5 and q_8, reading $\rho\lambda$ and pushing a 2, or reading μ and popping a 2, so $p_3\mu \in \mathcal{L}_{1,\infty}$. It follows that w has the bad prefix $u\rho\lambda_*(p_3\mu)\lambda\mu$, and so M correctly rejects it.

In the other four cases we have that $D_{\rho,\lambda}(p_2) = 1$ since p_2 ends at state q_1, q_4, q_5 or q_6, λ_* must be immediately followed by a letter ρ, and $p_2\lambda_*\rho$ ends at state q_6. Let $p_3 = \rho p_4$. Then $p_4\mu$ labels a path that starts at q_6, goes to q_8, then moves between states q_5 and q_8, reading $\rho\lambda$ and pushing a 2, or reading μ and popping a 2. So $p_4\mu \in \mathcal{L}_{1,\infty}$. It follows that w has the bad prefix $p_2\lambda_*\rho(p_4\mu)\lambda\mu$, and so M correctly rejects it. □

4 Obtaining the generating function

Theorem 11. *The sequence counting the number of permutations of each length in $\mathcal{P}$ has an algebraic generating function:*

$$\sum_{n\geqslant 0} c_n z^n = \frac{(1+q)\left(1+5q-q^2-q^3-(1-q)\sqrt{(1-q^2)(1-4q-q^2)}\right)}{8q}$$

where c_n is the number of permutations in $\mathcal{P}$ of length n, and $q \equiv q(z) = \frac{1-2z-\sqrt{1-4z}}{2z}$.

Proof. We convert the pushdown automaton given in the previous section to an unambiguous context-free language, following the standard procedure as described in Hopcroft and Ullman [14]. Theorem 10.12 of Hopcroft and Ullman guarantees that the grammar obtained from a deterministic pushdown automaton accepting on empty stack is $LR(0)$ and hence unambiguous.

We then apply the Chomsky and Schützenberger theorem, as outlined for example in [12] I.5.4, to obtain an algebraic generating function. Since each step in this procedure is constructive, we can find the generating function explicitly.

We start by converting the pushdown automaton to a grammar. See Theorem 5.4 [14] for full details.

Define a grammar with nonterminals S and $[q_i, j, q_k] = N_{i,j,k}$ for each pair of states q_i, q_k and stack symbol j. The nonterminal $[q_i, j, q_k]$ represents a path in the configuration space of the pushdown automaton starting at q_i with j on top of the stack and ending at some state q_k. The productions "fill out" these paths with subpaths according to the transitions that are possible.

The production rules are then defined as follows:

1. for each state q_i we have a production $S \to N_{00i}$,
2. for each transition $\delta(q_i, a, j) = \{(q_k, \varepsilon)\}$ with $a \in \{\$, \mu\}$, add a production $N_{ijk} = a$,
3. for each transition $\delta(q_i, \rho, j) = \{(q_k, l)\}$, add productions $N_{ijx} = \rho N_{klx}$ for $0 \leqslant x \leqslant 8$,
4. for each transition $\delta(q_i, \lambda, j) = \{(q_k, lm)\}$, add productions $N_{ijx} = \lambda N_{kly} N_{ymx}$ $0 \leqslant x, y \leqslant 8$.

This gives the following set of productions, where $0 \leqslant x, y \leqslant 8$:

$$
\begin{array}{lclclclclcl}
N_{000} & \to & \$ & \quad & N_{41x} & \to & \rho N_{21x} & \quad & N_{12x} & \to & \lambda N_{31y} N_{y2x} \\
N_{010} & \to & \mu & & N_{42x} & \to & \rho N_{22x} & & N_{20x} & \to & \lambda N_{51y} N_{y0x} \\
N_{020} & \to & \mu & & N_{11x} & \to & \rho N_{21x} & & N_{21x} & \to & \lambda N_{51y} N_{y1x} \\
N_{310} & \to & \mu & & N_{12x} & \to & \rho N_{22x} & & N_{22x} & \to & \lambda N_{51y} N_{y2x} \\
N_{414} & \to & \mu & & N_{31x} & \to & \rho N_{61x} & & N_{40x} & \to & \lambda N_{31y} N_{y0x} \\
N_{424} & \to & \mu & & N_{40x} & \to & \rho N_{20x} & & N_{41x} & \to & \lambda N_{31y} N_{y1x} \\
N_{514} & \to & \mu & & N_{51x} & \to & \rho N_{81x} & & N_{42x} & \to & \lambda N_{31y} N_{y2x}
\end{array}
$$

$$
\begin{array}{lclclclclcl}
N_{525} & \to & \mu & \quad & N_{52x} & \to & \rho N_{82x} & \quad & N_{51x} & \to & \lambda N_{71y} N_{y1x} \\
N_{00x} & \to & \rho N_{10x} & & N_{61x} & \to & \rho N_{81x} & & N_{52x} & \to & \lambda N_{71y} N_{y2x} \\
N_{01x} & \to & \rho N_{11x} & & N_{71x} & \to & \rho N_{61x} & & N_{61x} & \to & \lambda N_{71y} N_{y1x} \\
N_{02x} & \to & \rho N_{12x} & & N_{10x} & \to & \lambda N_{31y} N_{y0x} & & N_{81x} & \to & \lambda N_{52y} N_{y1x} \\
N_{10x} & \to & \rho N_{20x} & & N_{11x} & \to & \lambda N_{31y} N_{y1x} & & N_{82x} & \to & \lambda N_{52y} N_{y2x}
\end{array}
$$

We can reduce the size of the grammar description as follows. First, observe that the only productions that eliminate nonterminals (by generating \$ or μ) are of the form N_{*jk} for $k \in \{0, 4, 5\}$, and $j = 0$ implies $k = 0$. Since all productions with nonterminals on the right side have the form $N_{*ij} \to \rho N_{*ij}$ or $N_{*ij} \to \lambda N_{***} N_{*ij}$, it follows that any nonterminal N_{**k} with k not equal to $0, 4$ or 5 cannot be eliminated, so we can exclude them from the grammar.

Also, if we start a derivation with $S \to N_{00k}$ for $k \neq 0$, there will always be a nonterminal of the form N_{*0k} that cannot be eliminated. Therefore it suffices to make N_{000} the start nonterminal and remove all productions involving S.

Lastly, the resulting grammar contain nonterminals $N_{500}, N_{504}, N_{505}$ that will never produce a string of only terminals, since the configuration $(q_5, 0)$ is never realised (to reach q_5 the top of stack symbol is either 1 or 2. We modify the above grammar one step further by removing any production involving these nonterminals.

Taking these factors into consideration, and collecting productions with the same left side together we obtain the following grammar:

$$
\begin{array}{rcl}
N_{000} & \to & \$ \mid \rho N_{100}, \\
N_{004} & \to & \rho N_{104}, \\
N_{005} & \to & \rho N_{105}, \\
N_{010} & \to & \mu \mid \rho N_{110}, \\
N_{014} & \to & \rho N_{114}, \\
N_{015} & \to & \rho N_{115}, \\
N_{020} & \to & \mu \mid \rho N_{120}, \\
N_{024} & \to & \rho N_{124}, \\
N_{025} & \to & \rho N_{125}, \\
N_{100} & \to & \rho N_{200} \mid \lambda N_{310} N_{000} \mid \lambda N_{314} N_{400}, \\
N_{104} & \to & \rho N_{204} \mid \lambda N_{310} N_{004} \mid \lambda N_{314} N_{404}, \\
N_{105} & \to & \rho N_{205} \mid \lambda N_{310} N_{005} \mid \lambda N_{314} N_{405}, \\
N_{110} & \to & \rho N_{210} \mid \lambda N_{310} N_{010} \mid \lambda N_{314} N_{410} \mid \lambda N_{315} N_{510}, \\
N_{114} & \to & \rho N_{214} \mid \lambda N_{310} N_{014} \mid \lambda N_{314} N_{414} \mid \lambda N_{315} N_{514}, \\
N_{115} & \to & \rho N_{215} \mid \lambda N_{310} N_{015} \mid \lambda N_{314} N_{415} \mid \lambda N_{315} N_{515}, \\
N_{120} & \to & \rho N_{220} \mid \lambda N_{310} N_{020} \mid \lambda N_{314} N_{420} \mid \lambda N_{315} N_{520}, \\
N_{124} & \to & \rho N_{224} \mid \lambda N_{310} N_{024} \mid \lambda N_{314} N_{424} \mid \lambda N_{315} N_{524}, \\
N_{125} & \to & \rho N_{225} \mid \lambda N_{310} N_{025} \mid \lambda N_{314} N_{425} \mid \lambda N_{315} N_{525}, \\
N_{200} & \to & \lambda N_{510} N_{000} \mid \lambda N_{514} N_{400}, \\
N_{204} & \to & \lambda N_{510} N_{004} \mid \lambda N_{514} N_{404}, \\
N_{205} & \to & \lambda N_{510} N_{005} \mid \lambda N_{514} N_{405}, \\
N_{210} & \to & \lambda N_{510} N_{010} \mid \lambda N_{514} N_{410} \mid \lambda N_{515} N_{510}, \\
N_{214} & \to & \lambda N_{510} N_{014} \mid \lambda N_{514} N_{414} \mid \lambda N_{515} N_{514}, \\
N_{215} & \to & \lambda N_{510} N_{015} \mid \lambda N_{514} N_{415} \mid \lambda N_{515} N_{515}, \\
N_{220} & \to & \lambda N_{510} N_{020} \mid \lambda N_{514} N_{420} \mid \lambda N_{515} N_{520}, \\
N_{224} & \to & \lambda N_{510} N_{024} \mid \lambda N_{514} N_{424} \mid \lambda N_{515} N_{524}, \\
N_{225} & \to & \lambda N_{510} N_{025} \mid \lambda N_{514} N_{425} \mid \lambda N_{515} N_{525}, \\
N_{310} & \to & \mu \mid \rho N_{610}, \\
N_{314} & \to & \rho N_{614}, \\
N_{315} & \to & \rho N_{615}, \\
N_{400} & \to & \rho N_{200} \mid \lambda N_{310} N_{000} \mid \lambda N_{314} N_{400}, \\
N_{404} & \to & \rho N_{204} \mid \lambda N_{310} N_{004} \mid \lambda N_{314} N_{404}, \\
N_{405} & \to & \rho N_{205} \mid \lambda N_{310} N_{005} \mid \lambda N_{314} N_{405}, \\
N_{410} & \to & \rho N_{210} \mid \lambda N_{310} N_{010} \mid \lambda N_{314} N_{410} \mid \lambda N_{315} N_{510}, \\
N_{414} & \to & \mu \mid \rho N_{214} \mid \lambda N_{310} N_{014} \mid \lambda N_{314} N_{414} \mid \lambda N_{315} N_{514}, \\
N_{415} & \to & \rho N_{215} \mid \lambda N_{310} N_{015} \mid \lambda N_{314} N_{415} \mid \lambda N_{315} N_{515}, \\
N_{420} & \to & \rho N_{220} \mid \lambda N_{310} N_{020} \mid \lambda N_{314} N_{420} \mid \lambda N_{315} N_{520}, \\
N_{424} & \to & \mu \mid \rho N_{224} \mid \lambda N_{310} N_{024} \mid \lambda N_{314} N_{424} \mid \lambda N_{315} N_{524}, \\
N_{425} & \to & \rho N_{225} \mid \lambda N_{310} N_{025} \mid \lambda N_{314} N_{425} \mid \lambda N_{315} N_{525},
\end{array}
$$

$$
\begin{array}{rcl}
N_{510} & \to & \rho N_{810} \mid \lambda N_{710}N_{010} \mid \lambda N_{714}N_{410} \mid \lambda N_{715}N_{510}, \\
N_{514} & \to & \mu \mid \rho N_{814} \mid \lambda N_{710}N_{014} \mid \lambda N_{714}N_{414} \mid \lambda N_{715}N_{514}, \\
N_{515} & \to & \rho N_{815} \mid \lambda N_{710}N_{015} \mid \lambda N_{714}N_{415} \mid \lambda N_{715}N_{515}, \\
N_{520} & \to & \rho N_{820} \mid \lambda N_{710}N_{020} \mid \lambda N_{714}N_{420} \mid \lambda N_{715}N_{520}, \\
N_{524} & \to & \rho N_{824} \mid \lambda N_{710}N_{024} \mid \lambda N_{714}N_{424} \mid \lambda N_{715}N_{524}, \\
N_{525} & \to & \mu \mid \rho N_{825} \mid \lambda N_{710}N_{025} \mid \lambda N_{714}N_{425} \mid \lambda N_{715}N_{525}, \\
N_{610} & \to & \rho N_{810} \mid \lambda N_{710}N_{010} \mid \lambda N_{714}N_{410} \mid \lambda N_{715}N_{510}, \\
N_{614} & \to & \rho N_{814} \mid \lambda N_{710}N_{014} \mid \lambda N_{714}N_{414} \mid \lambda N_{715}N_{514}, \\
N_{615} & \to & \rho N_{815} \mid \lambda N_{710}N_{015} \mid \lambda N_{714}N_{415} \mid \lambda N_{715}N_{515}, \\
N_{710} & \to & \rho N_{610}, \\
N_{714} & \to & \rho N_{614}, \\
N_{715} & \to & \rho N_{615}, \\
N_{810} & \to & \lambda N_{520}N_{010} \mid \lambda N_{524}N_{410} \mid \lambda N_{525}N_{510}, \\
N_{814} & \to & \lambda N_{520}N_{014} \mid \lambda N_{524}N_{414} \mid \lambda N_{525}N_{514}, \\
N_{815} & \to & \lambda N_{520}N_{015} \mid \lambda N_{524}N_{415} \mid \lambda N_{525}N_{515}, \\
N_{820} & \to & \lambda N_{520}N_{020} \mid \lambda N_{524}N_{420} \mid \lambda N_{525}N_{520}, \\
N_{824} & \to & \lambda N_{520}N_{024} \mid \lambda N_{524}N_{424} \mid \lambda N_{525}N_{524}, \\
N_{825} & \to & \lambda N_{520}N_{025} \mid \lambda N_{524}N_{425} \mid \lambda N_{525}N_{525}.
\end{array}
$$

The next step is to convert nonterminals to generating functions, terminals to z and productions to equations, as described in [12] I.5.4.

$$
\begin{array}{rcl}
f_{000} & = & z + zf_{100}, \\
f_{004} & = & zf_{104}, \\
f_{005} & = & zf_{105}, \\
f_{010} & = & z + zf_{110}, \\
f_{014} & = & zf_{114}, \\
f_{015} & = & zf_{115}, \\
f_{020} & = & z + zf_{120}, \\
f_{024} & = & zf_{124}, \\
f_{025} & = & zf_{125}, \\
f_{100} & = & zf_{200} + zf_{310}f_{000} + zf_{314}f_{400}, \\
f_{104} & = & zf_{204} + zf_{310}f_{004} + zf_{314}f_{404}, \\
f_{105} & = & zf_{205} + zf_{310}f_{005} + zf_{314}f_{405}, \\
f_{110} & = & zf_{210} + zf_{310}f_{010} + zf_{314}f_{410} + zf_{315}f_{510}, \\
f_{114} & = & zf_{214} + zf_{310}f_{014} + zf_{314}f_{414} + zf_{315}f_{514}, \\
f_{115} & = & zf_{215} + zf_{310}f_{015} + zf_{314}f_{415} + zf_{315}f_{515}, \\
f_{120} & = & zf_{220} + zf_{310}f_{020} + zf_{314}f_{420} + zf_{315}f_{520}, \\
f_{124} & = & zf_{224} + zf_{310}f_{024} + zf_{314}f_{424} + zf_{315}f_{524}, \\
f_{125} & = & zf_{225} + zf_{310}f_{025} + zf_{314}f_{425} + zf_{315}f_{525},
\end{array}
$$

$$
\begin{aligned}
f_{200} &= zf_{510}f_{000} + zf_{514}f_{400},\\
f_{204} &= zf_{510}f_{004} + zf_{514}f_{404},\\
f_{205} &= zf_{510}f_{005} + zf_{514}f_{405},\\
f_{210} &= zf_{510}f_{010} + zf_{514}f_{410} + zf_{515}f_{510},\\
f_{214} &= zf_{510}f_{014} + zf_{514}f_{414} + zf_{515}f_{514},\\
f_{215} &= zf_{510}f_{015} + zf_{514}f_{415} + zf_{515}f_{515},\\
f_{220} &= zf_{510}f_{020} + zf_{514}f_{420} + zf_{515}f_{520},\\
f_{224} &= zf_{510}f_{024} + zf_{514}f_{424} + zf_{515}f_{524},\\
f_{225} &= zf_{510}f_{025} + zf_{514}f_{425} + zf_{515}f_{525},\\
f_{310} &= z + zf_{610},\\
f_{314} &= zf_{614},\\
f_{315} &= zf_{615},\\
f_{400} &= zf_{200} + zf_{310}f_{000} + zf_{314}f_{400},\\
f_{404} &= zf_{204} + zf_{310}f_{004} + zf_{314}f_{404},\\
f_{405} &= zf_{205} + zf_{310}f_{005} + zf_{314}f_{405},\\
f_{410} &= zf_{210} + zf_{310}f_{010} + zf_{314}f_{410} + zf_{315}f_{510},\\
f_{414} &= z + zf_{214} + zf_{310}f_{014} + zf_{314}f_{414} + zf_{315}f_{514},\\
f_{415} &= zf_{215} + zf_{310}f_{015} + zf_{314}f_{415} + zf_{315}f_{515},\\
f_{420} &= zf_{220} + zf_{310}f_{020} + zf_{314}f_{420} + zf_{315}f_{520},\\
f_{424} &= z + zf_{224} + zf_{310}f_{024} + zf_{314}f_{424} + zf_{315}f_{524},\\
f_{425} &= zf_{225} + zf_{310}f_{025} + zf_{314}f_{425} + zf_{315}f_{525},\\
f_{510} &= zf_{810} + zf_{710}f_{010} + zf_{714}f_{410} + zf_{715}f_{510},\\
f_{514} &= z + zf_{814} + zf_{710}f_{014} + zf_{714}f_{414} + zf_{715}f_{514},\\
f_{515} &= zf_{815} + zf_{710}f_{015} + zf_{714}f_{415} + zf_{715}f_{515},\\
f_{520} &= zf_{820} + zf_{710}f_{020} + zf_{714}f_{420} + zf_{715}f_{520},\\
f_{524} &= zf_{824} + zf_{710}f_{024} + zf_{714}f_{424} + zf_{715}f_{524},\\
f_{525} &= z + zf_{825} + zf_{710}f_{025} + zf_{714}f_{425} + zf_{715}f_{525},\\
f_{610} &= zf_{810} + zf_{710}f_{010} + zf_{714}f_{410} + zf_{715}f_{510},\\
f_{614} &= zf_{814} + zf_{710}f_{014} + zf_{714}f_{414} + zf_{715}f_{514},\\
f_{615} &= zf_{815} + zf_{710}f_{015} + zf_{714}f_{415} + zf_{715}f_{515},\\
f_{710} &= zf_{610},\\
f_{714} &= zf_{614},\\
f_{715} &= zf_{615},\\
f_{810} &= zf_{520}f_{010} + zf_{524}f_{410} + zf_{525}f_{510},\\
f_{814} &= zf_{520}f_{014} + zf_{524}f_{414} + zf_{525}f_{514},\\
f_{815} &= zf_{520}f_{015} + zf_{524}f_{415} + zf_{525}f_{515},\\
f_{820} &= zf_{520}f_{020} + zf_{524}f_{420} + zf_{525}f_{520},\\
f_{824} &= zf_{520}f_{024} + zf_{524}f_{424} + zf_{525}f_{524},\\
f_{825} &= zf_{520}f_{025} + zf_{524}f_{425} + zf_{525}f_{525}.
\end{aligned}
$$

Using Maple (version 14) we can solve to obtain an expression for the algebraic generating function $f_{000}(z)$, which counts the number of words in $\mathcal{L}\$$ of each length. Since words in $\mathcal{L}\$$ of length $3n+1$ are in bijection with permutations in $\mathcal{P}$ of length n, the generating function $\sum_{n \geqslant 0} c_n t^n$ where c_n is the number of permutations of length n in $\mathcal{P}$ is obtained by dividing f_{000} by z and substituting $z^3 = t$. $\square$

From the expression for the generating function we can easily obtain the first few terms of the sequence:

$1 + z + 2z^2 + 6z^3 + 24z^4 + 114z^5 + 592z^6 + 3216z^7 + 17904z^8 + 101198z^9 + 578208z^{10} + 3332136z^{11} + 19343408z^{12} + \dots$.

We can also use standard analytic combinatorial methods [12] to deduce the asymptotic growth of the number of such permutations:

$$c_n \sim \frac{\sqrt{25 - 11\sqrt{5}}}{2\sqrt{\pi n^3}} \cdot (2 + 2\sqrt{5})^n \cdot \left(1 + O(n^{-1})\right).$$

Acknowledgements

The bulk of this paper is the result of a University of Newcastle summer vacation project undertaken by the second author under supervision of the first. Research was supported by the Australian Research Council (ARC) grant FT110100178, and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] M. H. Albert, M. D. Atkinson, and N. Ruškuc. Regular closed sets of permutations. *Theoret. Comput. Sci.*, 306(1-3):85–100, 2003.

[2] M. H. Albert, M. Elder, A. Rechnitzer, P. Westcott, and M. Zabrocki. On the Stanley-Wilf limit of 4231-avoiding permutations and a conjecture of Arratia. *Adv. in Appl. Math.*, 36(2):96–105, 2006.

[3] Michael Albert and Mireille Bousquet-Mélou. Permutations sortable by two stacks in parallel and quarter plane walks. *Eur. J. Comb.*, 43:131–164, 2015.

[4] Michael H. Albert, M. D. Atkinson, and Vincent Vatter. Counting 1324, 4231-avoiding permutations. *Electron. J. Combin.*, 16(1):#R136, 2009.

[5] Michael H. Albert, Steve Linton, and Nik Ruškuc. The insertion encoding of permutations. *Electron. J. Combin.*, 12:#R47, 2005.

[6] M. D. Atkinson, M. J. Livesey, and D. Tulley. Permutations generated by token passing in graphs. *Theoret. Comput. Sci.*, 178(1-2):103–118, 1997.

[7] M. D. Atkinson, M. M. Murphy, and N. Ruškuc. Sorting with two ordered stacks in series. *Theoret. Comput. Sci.*, 289(1):205–223, 2002.

[8] Miklós Bóna. Exact enumeration of 1342-avoiding permutations: a close link with labeled trees and planar maps. *J. Combin. Theory Ser. A*, 80(2):257–272, 1997.

[9] Andrew R. Conway and Anthony J. Guttmann. On 1324-avoiding permutations. *Adv. in Appl. Math.*, 64:50–69, 2015.

[10] M. Elder. Pattern avoiding permutations are context-sensitive, 2004. arXiv:math/0412019

[11] Murray Elder. Permutations generated by a stack of depth 2 and an infinite stack in series. *Electron. J. Combin.*, 13(1):#R68, 2006.

[12] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, Cambridge, 2009.

[13] Ira M. Gessel. Symmetric functions and P-recursiveness. *J. Combin. Theory Ser. A*, 53(2):257–285, 1990.

[14] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. Addison-Wesley Series in Computer Science.

[15] Donald E. Knuth. *The art of computer programming. Volume 3*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1973. Sorting and searching, Addison-Wesley Series in Computer Science and Information Processing.

[16] Adeline Pierrot and Dominique Rossin. 2-stack sorting is polynomial. In *31st International Symposium on Theoretical Aspects of Computer Science*, volume 25 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 614–626. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2014.

[17] Rodica Simion and Frank W. Schmidt. Restricted permutations. *European J. Combin.*, 6(4):383–406, 1985.

[18] Wikipedia. Enumerations of specific permutation classes — Wikipedia, the free encyclopedia, 2014. http://en.wikipedia.org/wiki/Enumerations_of_specific_permutation_classes Accessed 5 July 2014.