Visibly Rational Expressions

Laura Bozzelli[1] **and César Sánchez**[2]

1 **Technical University of Madrid (UPM), Madrid, Spain**
`laura.bozzelli@fi.upm.es`

2 **IMDEA Software Institute, Madrid, Spain &**
Institute for Applied Physics, CSIC, Madrid, Spain
`cesar.sanchez@imdea.org`

Abstract

Regular Expressions (RE) are an algebraic formalism for expressing regular languages, widely used in string search and as a specification language in verification. In this paper we introduce and investigate Visibly Rational Expressions (VRE), an extension of RE for the well-known class of Visibly Pushdown Languages (VPL). We show that VRE capture the class of VPL. Moreover, we identify an *equally expressive* fragment of VRE which admits a quadratic time compositional translation into the automata acceptors of VPL. We also prove that, for this fragment, universality, inclusion and language equivalence are EXPTIME-complete. Finally, we provide an extension of VRE for VPL over infinite words.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Visibly Pushdown Languages, Context-free specifications, Regular expressions, Algebraic characterization

1 Introduction

Visibly Pushdown Languages (VPL), introduced by Alur et al. [4, 5], represent a robust and widely investigated subclass of context-free languages which includes strictly the class of regular languages. A VPL consists of *nested words*, that is words over an alphabet (pushdown alphabet) which is partitioned into three disjoint sets of calls, returns, and internal symbols. This partition induces a nested hierarchical structure in a given word obtained by associating to each call the corresponding matching return (if any) in a well-nested manner. VPL are accepted by Visibly Pushdown Automata (VPA), a subclass of pushdown automata which push onto the stack only when a call is read, pops the stack only at returns, and do not use the stack on reading internal symbols. Hence, the input controls the kind of operations permissible on the stack, and thus the stack depth at every position [4]. This restriction makes the class of VPL very similar in tractability and robustness to that of regular languages. In particular, VPL are closed under intersection, union, complementation, renaming, concatenation and Kleene closure [4]. Moreover, VPA (over finite words) are determinizable, and decision problems like universality, equivalence and inclusion – which are undecidable for context-free languages – become EXPTIME-complete for VPL. Furthermore, various alternative and constructive characterizations of VPL have been given in terms of operational and descriptive formalisms: logical characterizations by standard MSO over nested words extended with a binary matching-predicate (MSO_μ) [4] or by fixpoint logics [10], a context-free grammar based characterization [4], alternating automata based characterizations [6, 10], and a congruence-based characterization [3].

The theory of VPL has relevant applications in the formal verification and synthesis of sequential recursive programs with finite data types modeled by pushdown systems [7, 5, 2, 18]. Runs in these programs can be seen as nested words capturing the nested calling

LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

structure, where the call to and return from procedures capture the nesting. Additionally, VPL have applications in the streaming processing of semi-structured data, such as XML documents, where each open-tag is matched with a closing-tag in a well-nested manner [20, 16, 19, 1, 21]. Examples include type-checking (validation) and dynamic typing of XML documents against schema specifications [16], and evaluation of MSO_μ queries on streaming XML documents [19].

Contribution. Regular Expressions [15, 14] (RE) are an algebraic formalism for describing regular languages. RE are widely adopted as a descriptive specification language, for example in string search [22], and for extensions of temporal logics for hardware model checking [13, 17]. In this paper, we introduce and investigate a similar algebraic formalism for the class of VPL, that we call *Visibly Regular Expressions* (VRE). VRE extend RE by adding two novel non-regular operators which are parameterized by an internal action: (1) the binary *Minimally Well-Matched Substitution* operator (M-substitution for short), which allows to substitute occurrences of the designated internal action by *minimally well-matched* (*MWM*) words;[1] (2) and the unary *Strict Mimimally Well-Matched Closure* operator (S-closure for short), which corresponds to the (unbounded) iteration of the M-substitution operation. We also consider a third operator which can be expressed in terms of M-substitution and S-closure. The class of *pure* VRE is obtained by disallowing the explicit use of this operator. Intuitively, M-substitution and S-closure, when applied to languages $\mathcal{L}$ of *MWM* words, correspond to classical tree language concatenation and Kleene closure applied to the tree language encoding of the (nested) word languages $\mathcal{L}$ (in accordance with the standard encoding of *MWM* words by ordered unranked finite trees [1]).

Our results are as follows. First, we establish that VRE capture exactly the class of VPL. Like the classical Kleene theorem [15], the translation from automata (VPA) to expressions (VRE) involves a singly exponential blow-up. For the converse direction (from VRE to VPA), the proposed construction requires again single exponential time (it is an open question if this exponential blow-up can be avoided). On the other hand, we show that pure VRE (which are equivalent to unrestricted VRE) can be compositionally converted in quadratic time into equivalent VPA. The key of this translation is given by a novel subclass of VPA. Next, we prove that universality, inclusion, and language equivalence for pure VRE are EXPTIME-complete. Finally, we also provide an algebraic characterization of VPL over infinite words.

A potential application of our algebraic formalism is as a schema specification language for semi-structured data such as XML documents. In fact, usually, XML schema specifications are context-free grammars and their derivation trees give the tree representation of the associated sets of XML documents. So, these specifications are typically compiled into tree automata. However, it has been shown [16, 19, 1, 21] that VPA are often more natural (and sometime exponentially more succinct) than tree automata, and moreover preferable in the streaming processing of XML documents.

Related Work. Another algebraic characterization of VPL has been given in [8], where regular expressions are extended with an infinite family of operators, which are implicit least fix-points. In fact, these operators encode in a linear way a subclass of context-free grammars. In comparison, we introduce just two operators which make our formalism really more lightweight and intuitive to use. In [12] a characterization of VPL is given by morphisms to suitable algebraic structures. Moreover, [12] introduces an extension of the Kleene-closure free fragment of regular expressions obtained by adding a variant of

[1] *MWM* words are words whose first symbol is a call and whose last symbol is the matching return.

our substitution operator, and shows that this extension captures exactly the first-order fragment of MSO_μ over well-matched words.

2 Visibly pushdown languages

In this section, we recall the class of visibly pushdown automata and visibly pushdown languages [4].

Pushdown Alphabet. A *pushdown alphabet* is a tuple $\widetilde{\Sigma} = \langle \Sigma_{call}, \Sigma_{ret}, \Sigma_{int} \rangle$ consisting of three disjoint finite alphabets: Σ_{call} is a finite set of *calls*, Σ_{ret} is a finite set of *returns*, and Σ_{int} is a finite set of internal actions. For any such $\widetilde{\Sigma}$, the *support* of $\widetilde{\Sigma}$ is $\Sigma = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$. We will use $c, c_1, c_i, \ldots$ for elements of Σ_{call}, $r, r_1, r_i, \ldots$ for elements of Σ_{ret}, $\square, \square_1, \square_j, \ldots$ for elements of Σ_{int}, and $\sigma, \sigma_1, \sigma_i, \ldots$ for arbitrary elements of Σ.

Visibly Pushdown Automata and Visibly Pushdown Languages. A *Nondeterministic Visibly Pushdown Automaton on finite words* (NVPA) [4] *over* $\widetilde{\Sigma} = \langle \Sigma_{call}, \Sigma_{ret}, \Sigma_{int} \rangle$ is a tuple $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F \rangle$, where Q is a finite set of (control) states, $q_{in} \in Q$ is the initial state, Γ is a finite stack alphabet, $\Delta \subseteq (Q \times \Sigma_{call} \times Q \times \Gamma) \cup (Q \times \Sigma_{ret} \times (\Gamma \cup \{\bot\}) \times Q) \cup (Q \times \Sigma_{int} \times Q)$ is a transition relation (where $\bot \notin \Gamma$ is the special *stack bottom symbol*), and $F \subseteq Q$ is a set of accepting states. A transition of the form $(q, c, q', \gamma) \in Q \times \Sigma_{call} \times Q \times \Gamma$ is a push transition, where on reading the call c, the symbol $\gamma \neq \bot$ is pushed onto the stack and the control changes from q to q'. A transition of the form $(q, r, \gamma, q') \in Q \times \Sigma_{ret} \times (\Gamma \cup \{\bot\}) \times Q$ is a pop transition, where on reading the return r, γ is read from the top of the stack and popped, and the control changes from q to q' (if the top of the stack is $\bot$, then it is read but not popped). Finally, on reading an internal action $\square$, $\mathcal{P}$ can choose only transitions of the form $(q, \square, q')$ which do not use the stack.

A configuration of $\mathcal{P}$ is a pair (q, β), where $q \in Q$ and $\beta \in \Gamma^* \cdot \{\bot\}$ is a stack content. A run π of $\mathcal{P}$ over a finite word $\sigma_1 \ldots \sigma_{n-1} \in \Sigma^*$ is a finite sequence of the form $\pi = (q_1, \beta_1) \xrightarrow{\sigma_1} (q_2, \beta_2) \ldots \xrightarrow{\sigma_{n-1}} (q_n, \beta_n)$ such that (q_i, β_i) is a configuration for all $1 \leq i \leq n$, and the following holds for all $1 \leq i \leq n-1$:

Push If σ_i is a call, then for some $\gamma \in \Gamma$, $(q_i, \sigma_i, q_{i+1}, \gamma) \in \Delta$ and $\beta_{i+1} = \gamma \cdot \beta_i$.

Pop If σ_i is a return, then for some $\gamma \in \Gamma \cup \{\bot\}$, $(q_i, \sigma_i, \gamma, q_{i+1}) \in \Delta$, and *either* $\gamma \neq \bot$ and $\beta_i = \gamma \cdot \beta_{i+1}$, *or* $\gamma = \bot$ and $\beta_i = \beta_{i+1} = \bot$.

Internal If σ_i is an internal action, then $(q_i, \sigma_i, q_{i+1}) \in \Delta$ and $\beta_{i+1} = \beta_i$.

For all $1 \leq i \leq j \leq n$, the subsequence of π given by $\pi_{ij} = (q_i, \beta_i) \xrightarrow{\sigma_i} \ldots \xrightarrow{\sigma_{j-1}} (q_j, \beta_j)$ is called *subrun of* π (note that π_{ij} is a run over $\sigma_i \ldots \sigma_{j-1}$). The run π is *initialized* if $q_1 = q_{in}$ and $\beta_1 = \bot$. Moreover, the run π is *accepting* if the last state is accepting, that is, if $q_n \in F$. For two configurations (q, β) and (q', β') and a finite word $w \in \Sigma^*$, we write $(q, \beta) \xrightarrow{w} (q', \beta')$ to mean that there is a run of $\mathcal{P}$ over w starting at (q, β) and leading to (q', β'). The language of $\mathcal{P}$, $\mathcal{L}(\mathcal{P})$, is the set of finite words $w \in \Sigma^*$ such that there is an initialized accepting run of $\mathcal{P}$ on w. A language of finite words $\mathcal{L} \subseteq \Sigma^*$ is a *visibly pushdown language* (VPL) *with respect to* $\widetilde{\Sigma}$ if there is a NVPA $\mathcal{P}$ over $\widetilde{\Sigma}$ such that $\mathcal{L} = \mathcal{L}(\mathcal{P})$.

We also consider *Visibly Pushdown Automata on infinite words* (ω-NVPA). Formally, a Büchi ω-NVPA *over* $\widetilde{\Sigma}$ [4] is a tuple $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F \rangle$, where $Q, q_{in}, \Gamma, \Delta$, and F are defined as for NVPA over $\widetilde{\Sigma}$. A run π over an infinite word $\sigma_1 \sigma_2 \ldots \in \Sigma^\omega$ is an infinite sequence $\pi = (q_1, \beta_1) \xrightarrow{\sigma_1} (q_2, \beta_2) \ldots$ that is defined using the natural extension of the definition of runs on finite words. The run is *accepting* if for infinitely many $i \geq 1$, $q_i \in F$. The notions of initialized run and (finite) subrun are defined as for NVPA. The ω-language of $\mathcal{P}$, $\mathcal{L}(\mathcal{P})$, is the set of infinite words $w \in \Sigma^\omega$ such that there is an initialized accepting run of

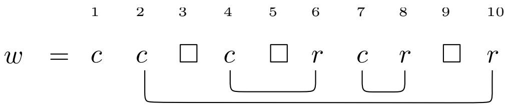
$\mathcal{P}$ on w. An ω-language $\mathcal{L} \subseteq \Sigma^\omega$ is an *ω-visibly pushdown language* (ω-VPL) *with respect to* $\widetilde{\Sigma}$ if there is a Büchi ω-NVPA $\mathcal{P}$ over $\widetilde{\Sigma}$ such that $\mathcal{L} = \mathcal{L}(P)$.

Matched calls and returns. Fix a pushdown alphabet $\widetilde{\Sigma} = \langle \Sigma_{call}, \Sigma_{ret}, \Sigma_{int} \rangle$. For a finite or infinite word w over Σ, $|w|$ is the length of w (we set $|w| = \omega$ if w is infinite). For all $1 \leq i \leq |w|$, $w(i)$ is the i^{th} symbol of w. A position $1 \leq i \leq |w|$ of w is a call (resp., return, internal) position if $w(i) \in \Sigma_{call}$ (resp., $w(i) \in \Sigma_{ret}$, $w(i) \in \Sigma_{int}$).

The set $WM(\widetilde{\Sigma})$ of *well-matched words* is the subset of Σ^* inductively defined as follows: (i) $\epsilon \in WM(\widetilde{\Sigma})$, where ϵ denotes the empty string, (ii) $\Box \cdot w \in WM(\widetilde{\Sigma})$ if $\Box \in \Sigma_{int}$ and $w \in WM(\widetilde{\Sigma})$, and (iii) $c \cdot w \cdot r \cdot w' \in WM(\widetilde{\Sigma})$ if $c \in \Sigma_{call}$, $r \in \Sigma_{ret}$, and $w, w' \in WM(\widetilde{\Sigma})$.

Let i be a call position of a word w. If there is $j > i$ such that j is a return position of w and $w(i+1) \dots w(j-1)$ is a well-matched word (note that j is uniquely determined if it exists), we say that j is the *matching return* of i along w, and i is the *matching call* of j along w. The set $MWM(\widetilde{\Sigma})$ of *minimally well-matched words* is the set of well-matched words of the form $c \cdot w \cdot r$ such that c is a call, r is a return, and w is well-matched (note that r corresponds to the matching return of c). For a language $\mathcal{L} \subseteq \Sigma^*$, we define $MWM(\mathcal{L}) \overset{\text{def}}{=} \mathcal{L} \cap MWM(\widetilde{\Sigma})$, that is the set of words in $\mathcal{L}$ which are minimally well-matched.

▶ **Example 1.** Let $\Sigma_{call} = \{c\}$, $\Sigma_{ret} = \{r\}$, and $\Sigma_{int} = \{\Box\}$. Consider the word w below. The word w is not well-matched, in particular, the call at position 1 has no matching return in w. Moreover, note that the subword $w[2] \dots w[10]$ is minimally well-matched.



3 Visibly Rational Expressions (VRE)

In this section, we introduce and investigate the class of *Visibly Rational Expressions* (VRE), an extension of regular expressions obtained by adding two novel non-regular operators: the binary M-substitution operator and the unary S-closure operator. First, we define the corresponding operations on languages of finite words and show that VPL are effectively closed under these operations. Then, in Subsection 3.1, we introduce VRE and establish their effective language equivalence to the class of NVPA.

Let us fix a pushdown alphabet $\widetilde{\Sigma} = \langle \Sigma_{call}, \Sigma_{ret}, \Sigma_{int} \rangle$. For two languages $\mathcal{L}, \mathcal{L}' \subseteq \Sigma^*$ of finite words on Σ, we use $\mathcal{L} \cdot \mathcal{L}'$ to denote the standard concatenation of $\mathcal{L}$ and $\mathcal{L}'$, and $\mathcal{L}^*$ to denote the standard Kleene closure of $\mathcal{L}$.

▶ **Definition 2** (M-substitution). *Let $w \in \Sigma^*$, $\Box \in \Sigma_{int}$, and $\mathcal{L} \subseteq \Sigma^*$. The M-substitution of $\Box$ by $\mathcal{L}$ in w*, denoted by $w \curvearrowleft_{\Box} \mathcal{L}$, is the language of finite words over Σ obtained by replacing occurrences of $\Box$ in w by minimally well-matched words in $\mathcal{L}$. Formally, $w \curvearrowleft_{\Box} \mathcal{L}$ is inductively defined as follows:

- $\varepsilon \curvearrowleft_{\Box} \mathcal{L} \overset{\text{def}}{=} \{\varepsilon\}$;
- $(\Box \cdot w') \curvearrowleft_{\Box} \mathcal{L} \overset{\text{def}}{=} \big(MWM(\mathcal{L}) \cdot (w' \curvearrowleft_{\Box} \mathcal{L})\big) \cup \big((\{\Box\} \cap \mathcal{L}) \cdot (w' \curvearrowleft_{\Box} \mathcal{L})\big)$
- $(\sigma \cdot w') \curvearrowleft_{\Box} \mathcal{L} \overset{\text{def}}{=} \{\sigma\} \cdot (w' \curvearrowleft_{\Box} \mathcal{L})$ for each $\sigma \in \Sigma \setminus \{\Box\}$.

For two languages $\mathcal{L}, \mathcal{L}' \subseteq \Sigma^*$ and $\Box \in \Sigma_{int}$, the *M-substitution of $\Box$ by $\mathcal{L}'$ in $\mathcal{L}$*, written $\mathcal{L} \curvearrowleft_{\Box} \mathcal{L}'$, is defined as

$$\mathcal{L} \curvearrowleft_{\Box} \mathcal{L}' \overset{\text{def}}{=} \bigcup_{w \in \mathcal{L}} w \curvearrowleft_{\Box} \mathcal{L}'$$

Note that $\curvearrowleft_{\square}$ is associative. Moreover, if $\{\square\} \cap \mathcal{L} = \emptyset$, then $\{\square\} \curvearrowleft_{\square} \mathcal{L} = MWM(\mathcal{L})$.

▶ **Example 3.** Let $\Sigma_{call} = \{c_1, c_2\}$, $\Sigma_{ret} = \{r\}$, and $\Sigma_{int} = \{\square\}$. Let us consider the languages $\mathcal{L} = \{c_1^n \, \square \, \square \, r^n \mid n \geq 1\}$ and $\mathcal{L}' = \{c_2\}^* \cdot \{r\}^*$. Then $\mathcal{L} \curvearrowleft_{\square} \mathcal{L}'$ is given by $\{c_1^n \, c_2^m \, r^m \, c_2^k \, r^k \, r^n \mid n, m, k \geq 1\}$.

▶ **Definition 4** (M-closure and S-closure)**.** Given $\mathcal{L} \subseteq \Sigma^*$ and $\square \in \Sigma_{int}$, the M-closure of $\mathcal{L}$ through $\square$, denoted by $\mathcal{L}^{\curvearrowleft_{\square}}$, is defined as:

$$\mathcal{L}^{\curvearrowleft_{\square}} \overset{\text{def}}{=} \bigcup_{n \geq 0} \mathcal{L} \underbrace{\curvearrowleft_{\square} (\mathcal{L} \cup \{\square\}) \curvearrowleft_{\square} \ldots \curvearrowleft_{\square} (\mathcal{L} \cup \{\square\})}_{n \text{ occurrences of } \curvearrowleft_{\square}}.$$

The S-closure of $\mathcal{L}$ through $\square$, denoted by $\mathcal{L}^{\circlearrowleft_{\square}}$, is defined as $(MWM(\mathcal{L}))^{\curvearrowleft_{\square}}$. Note that $\mathcal{L}^{\circlearrowleft_{\square}}$ is contained in $MWM(\widetilde{\Sigma})$. The M-closure operator is a derived operator, since it can be expressed in terms of S-closure and M-substitution as follows:

$$\mathcal{L}^{\curvearrowleft_{\square}} = \mathcal{L} \curvearrowleft_{\square} (\mathcal{L}^{\circlearrowleft_{\square}} \cup \{\square\})$$

▶ **Example 5.** Let $\Sigma_{call} = \{c_1, c_2\}$, $\Sigma_{ret} = \{r_1, r_2\}$, and $\Sigma_{int} = \{\square\}$. Let us consider the languages $\mathcal{L} = \{\square, c_1 \, \square \, r_1, \, c_2 \, \square \, r_2\}$ and $\mathcal{L}' = \{c_1 \, r_1, \, c_2 \, r_2\}$. Then, $\mathcal{L}^{\curvearrowleft_{\square}} \curvearrowleft_{\square} \mathcal{L}' = \{c_{i_1} \, c_{i_2} \ldots c_{i_n} \, r_{i_n} \ldots r_{i_2} \, r_{i_1} \mid n \geq 1, \; i_1, \ldots, i_n \in \{1, 2\}\}$. One can easily show that there is no regular language $\mathcal{L}_{reg}$ such that $MWM(\mathcal{L}_{reg}) = \mathcal{L}^{\curvearrowleft_{\square}} \curvearrowleft_{\square} \mathcal{L}'$.

Now we show that VPL are closed under M-substitution, M-closure and S-closure.

▶ **Theorem 6.** *Let $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ and $\mathcal{P}' = \langle Q', q'_{in}, \Gamma', \Delta', F'\rangle$ be two NVPA over $\widetilde{\Sigma}$, and $\square \in \Sigma_{int}$. Then, one can construct in polynomial time:*

1. *an NVPA accepting $(\mathcal{L}(\mathcal{P}))^{\circlearrowleft_{\square}}$ with $|Q| + 2$ states and $|\Gamma| \cdot (|Q| + 2)$ stack symbols.*
2. *an NVPA accepting $\mathcal{L}(\mathcal{P}) \curvearrowleft_{\square} \mathcal{L}(\mathcal{P}')$ with $|Q| + |Q'|$ states and $|\Gamma| + |\Gamma'| \cdot (|Q| + 1)$ stack symbols.*
3. *an NVPA accepting $(\mathcal{L}(\mathcal{P}))^{\curvearrowleft_{\square}}$ with $2|Q| + 2$ states and $2|\Gamma| \cdot (|Q| + 1)$ stack symbols.*

Proof. Here, we sketch the construction of the NVPA for Condition 1. Fix an NVPA $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$. The construction consists of two steps. In the first step, we construct an NVPA $\mathcal{P}' = \langle Q', q'_{in}, \Gamma \cup \widehat{\Gamma}, \Delta', F'\rangle$ accepting $MWM(\mathcal{L}(\mathcal{P}))$, where $Q' \supseteq Q$, $|Q'| = |Q| + 2$, and $\widehat{\Gamma}$ is a fresh copy of Γ; moreover, each push transition from q'_{in} pushes onto the stack a symbol in $\widehat{\Gamma}$ and each internal transition is in Δ. In the second step, we construct an NVPA $\mathcal{P}''$ accepting $(\mathcal{L}(\mathcal{P}'))^{\circlearrowleft_{\square}}$ with $|Q'|$ states and stack alphabet $\Gamma \cup \widehat{\Gamma} \cup Q \times \widehat{\Gamma}$. Hence, the result follows. Here, we informally describe the construction of $\mathcal{P}''$. Essentially, $\mathcal{P}''$ simulates $\mathcal{P}'$ step by step, but when $\mathcal{P}'$ performs an internal transition of the form $(q, \square, q')$, then from the current state q, $\mathcal{P}''$ can choose to either process $\square$ as $\mathcal{P}'$, or recursively process instead some guessed word $w \in \mathcal{L}(\mathcal{P}')^{\circlearrowleft_{\square}}$ as follows. $\mathcal{P}''$ guesses a call c that is the initial symbol of w, and chooses a push transition $(q'_{in}, c, p, \gamma) \in \Delta'$ from the initial state of $\mathcal{P}'$ (the construction of $\mathcal{P}'$ ensures that $\gamma \in \widehat{\Gamma}$). In the same step, $\mathcal{P}''$ pushes onto the stack both γ and the target state $q' \in Q$ of the internal transition $(q, \square, q')$ of $\mathcal{P}'$, and moves to state p. This compound step allows $\mathcal{P}''$ to guarantee that when the stack is popped on reading the matching return of c (corresponding to the last symbol of the guessed word w), $\mathcal{P}''$ can restart the simulation of $\mathcal{P}'$ from the desired control state q'. Moreover, when the matching return of c is read, $\mathcal{P}''$ guarantee that the pair (q', γ) is popped from the stack if and only if from the current state of $\mathcal{P}''$, there is some pop transition of $\mathcal{P}'$ which pops γ and leads to some accepting state in F'. Note that since $\mathcal{P}'$ accepts only minimally well-matched words, the pair (q', γ) pushed onto the stack is eventually popped. ◀

3.1 VRE and Equivalence Between VRE and NVPA

▶ **Definition 7** (VRE). The syntax of VRE E over the pushdown alphabet $\widetilde{\Sigma}$ is defined as:

$$E := \emptyset \;\big|\; \varepsilon \;\big|\; \sigma \;\big|\; (E \cup E) \;\big|\; (E \cdot E) \;\big|\; E^* \;\big|\; (E \curvearrowleft_{\Box} E) \;\big|\; E^{\circlearrowleft_{\Box}} \;\big|\; E^{\curvearrowleft_{\Box}}$$

where $\sigma \in \Sigma$ and $\Box \in \Sigma_{int}$. A *pure* VRE is a VRE which does not contain occurrences of the M-closure operator ${}^{\curvearrowleft_{\Box}}$. A VRE E denotes a language of finite words over Σ, written $\mathcal{L}(E)$, which is defined in the obvious way as follows:

- $\mathcal{L}(\emptyset) = \emptyset$, $\mathcal{L}(\varepsilon) = \{\varepsilon\}$, and $\mathcal{L}(\sigma) = \{\sigma\}$ for each $\sigma \in \Sigma$;
- $\mathcal{L}(E_1 \cup E_2) = \mathcal{L}(E_1) \cup \mathcal{L}(E_2)$, $\mathcal{L}(E_1 \cdot E_2) = \mathcal{L}(E_1) \cdot \mathcal{L}(E_2)$, and $\mathcal{L}(E^*) = [\mathcal{L}(E)]^*$;
- $\mathcal{L}(E_1 \curvearrowleft_{\Box} E_2) = \mathcal{L}(E_1) \curvearrowleft_{\Box} \mathcal{L}(E_2)$, $\mathcal{L}(E^{\circlearrowleft_{\Box}}) = [\mathcal{L}(E)]^{\circlearrowleft_{\Box}}$, and $\mathcal{L}(E^{\curvearrowleft_{\Box}}) = [\mathcal{L}(E)]^{\curvearrowleft_{\Box}}$.

As usual, the size $|E|$ of a VRE E is the length of the string describing E.

▶ Remark. By Definition 4, the M-closure operator is a derived operator. Hence, pure VRE and unrestricted VRE capture the same class of languages.

It is known [1] that for the class of regular languages (over a pushdown alphabet), NVPA can be exponentially more succinct than nondeterministic finite-state automata (NFA). We establish an analogous result for VRE and regular expressions.

▶ **Theorem 8.** *There are a pushdown alphabet $\widetilde{\Sigma}$ and a family $\{\mathcal{L}_n\}_{n\geq 1}$ of regular languages over $\widetilde{\Sigma}$ such that for each $n \geq 1$, $\mathcal{L}_n$ can be denoted by a VRE of size $O(n)$ and every regular expression denoting $\mathcal{L}_n$ has size at least $2^{\Omega(n)}$.*

Proof. Let $\widetilde{\Sigma} = \langle \Sigma_{call}, \Sigma_{ret}, \{\Box\}\rangle$ with $\Sigma_{call} = \{c_1, c_2\}$ and $\Sigma_{ret} = \{r_1, r_2\}$. For $n \geq 1$, let $\mathcal{L}_n$ be the finite (hence, regular) language $\{c_{i_1} c_{i_2} \ldots c_{i_n} r_{i_n} \ldots r_{i_2} r_{i_1} \mid i_1, \ldots, i_n \in \{1, 2\}\}$. Evidently, $\mathcal{L}_n$ can be expressed by the VRE of size $O(n)$ given by $\underbrace{E \curvearrowleft_{\Box} E \curvearrowleft_{\Box} \ldots \curvearrowleft_{\Box} E}_{n-1 \text{ times}} \curvearrowleft_{\Box}$ $(c_1 \cdot r_1 \cup c_2 \cdot r_2)$, where $E = (c_1 \cdot \Box \cdot r_1 \cup c_2 \cdot \Box \cdot r_2)$. However, as shown in [1], any NFA accepting $\mathcal{L}_n$ requires at least 2^n states. Thus, since regular expressions can be converted in linear time into equivalent NFA, the result follows. ◀

In the following, we show that VRE and NVPA are effectively language equivalent. First, we recall the following known result [4].

▶ **Theorem 9** (From [4]). *Let $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ and $\mathcal{P}' = \langle Q', q'_{in}, \Gamma', \Delta', F'\rangle$ be two NVPA over $\widetilde{\Sigma}$. Then, one can construct in linear time:*

1. *an NVPA accepting $\mathcal{L}(\mathcal{P}) \cup \mathcal{L}(\mathcal{P}')$ (resp. $\mathcal{L}(\mathcal{P}) \cdot \mathcal{L}(\mathcal{P}')$) with $|Q| + |Q'|$ states and $|\Gamma| + |\Gamma'|$ stack symbols.*
2. *an NVPA accepting $[\mathcal{L}(\mathcal{P})]^*$ with $2|Q|$ states and $2|\Gamma|$ stack symbols.*

By Theorems 6 and 9, a given VRE can be effectively and compositionally translated into an equivalent NVPA. However, due to Condition 3 in Theorem 6 (concerning the M-closure operator) and Condition 2 in Theorem 9 (concerning the Kleene closure operator), the translation can involve a singly exponential blow-up. In the next section, we show that this exponential blow-up is due essentially to the presence of the M-closure operator (in particular, we propose a quadratic time translation of *pure* VRE into equivalent NVPA).

▶ **Corollary 10.** *Given a VRE E, one can construct in* singly exponential time *an NVPA accepting $\mathcal{L}(E)$.*

Now, we show that any NVPA can be converted into an equivalent VRE. First, we need some additional notation. Let $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ be an NVPA over a pushdown alphabet $\widetilde{\Sigma}$. Given $p, p' \in Q$, a *summary* of $\mathcal{P}$ from p to p' is a run π of $\mathcal{P}$ over some word $w \in MWM(\widetilde{\Sigma})$ from a configuration of the form (p, β) to a configuration of the form (p', β') for some stack contents β and β'. Observe that if π is such a run over $w \in MWM(\widetilde{\Sigma})$ from (p, β) to (p', β'), then $\beta' = \beta$ and the portion of the stack corresponding to β is never read in π. In particular, there is also a run of $\mathcal{P}$ from $(p, \bot)$ to $(p', \bot)$ over w which uses the same transitions used by π. Given a run π of $\mathcal{P}$ over some word w and $\mathcal{S} \subseteq Q \times Q$, we say that the run π uses only *sub-summaries* from $\mathcal{S}$ whenever for all $q, q' \in Q$, if there is subrun of π which is a summary from q to q', then $(q, q') \in \mathcal{S}$. Given a finite alphabet Λ disjoint from Σ, we denote by $\widetilde{\Sigma}_\Lambda$ the pushdown alphabet $\langle \Sigma_{call}, \Sigma_{ret}, \Sigma_{int} \cup \Lambda\rangle$ obtained by interpreting the elements in Λ as internal actions.

▶ **Theorem 11.** *Given an NVPA $\mathcal{P}$, one can construct in single exponential time a VRE E such that $\mathcal{L}(E) = \mathcal{L}(\mathcal{P})$.*

Proof. Let $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$. We construct a finite alphabet Λ disjoint from Σ and a VRE E over $\widetilde{\Sigma}_\Lambda$ denoting $\mathcal{L}(\mathcal{P})$; the additional symbols in Λ are used only as parameters for intermediate substitutions. The alphabet Λ is given by $\{\square_{pp'} \mid p, p' \in Q\}$. Moreover, we define $\mathcal{P}_\Lambda = \langle Q, q_{in}, \Gamma, \Delta_\Lambda, F\rangle$ as the NVPA over $\widetilde{\Sigma}_\Lambda$ obtained from $\mathcal{P}$ by adding for each $(p, p') \in Q \times Q$, the internal transition $(p, \square_{pp'}, p')$. Given $q, q' \in Q$, $\mathcal{S} \subseteq Q \times Q$, and $\Lambda' \subseteq \Lambda$, we define $R(q, q', \mathcal{S}, \Lambda')$ as the language of finite words w over $\Sigma_{\Lambda'}$ ($\Sigma_{\Lambda'}$ is the support of $\widetilde{\Sigma}_{\Lambda'}$) such that there is a run of $\mathcal{P}_\Lambda$ over w from $(q, \bot)$ to some configuration of the form (q', β) which uses only sub-summaries from $\mathcal{S}$.[2] By construction, $\mathcal{L}(\mathcal{P})$ is the union of the sets $R(q, q', Q \times Q, \emptyset)$ such that $q = q_{in}$ and $q' \in F$. Thus, the theorem follows from the fact that for all $q, q' \in Q$, $\mathcal{S} \subseteq Q \times Q$, and $\Lambda' \subseteq \Lambda$, the languages $R(q, q', \mathcal{S}, \Lambda')$ and $WM(R(q, q', \mathcal{S}, \Lambda'))$ can be effectively denoted by VRE of sizes singly exponential in the size of $\mathcal{P}$, where $WM(R(q, q', \mathcal{S}, \Lambda')) \overset{\text{def}}{=} R(q, q', \mathcal{S}, \Lambda') \cap WM(\widetilde{\Sigma})$. The proof of this fact proceeds by induction on the cardinality of the finite set $\mathcal{S}$.

Base case: $\mathcal{S} = \emptyset$. The proof of the base case is simple. In particular, the languages $R(q, q', \emptyset, \Lambda')$ and $WM(R(q, q', \emptyset, \Lambda'))$ are regular.

Induction step: $\mathcal{S} = \mathcal{S}' \cup \{(p, p')\}$ with $(p, p') \notin \mathcal{S}'$. Let $P_{p \to p'}$ be the set:

$$\{(s, c, r, s') \in Q \times \Sigma_{call} \times \Sigma_{ret} \times Q \mid \textit{ there is } \gamma \in \Gamma.(p, c, s, \gamma), (s', r, \gamma, p') \in \Delta\}$$

So, $P_{p \to p'}$ is the set of tuples (s, c, r, s') such that there is a push transition from p to s reading the call c and a matching pop transition from s' to p' reading r. Moreover, let $S(p, p', \mathcal{S}' \cup \{(p, p')\}, \Lambda')$ be the language over $\Sigma_{\Lambda'}$ defined as follows:

$$\begin{aligned}
&S(p, p', \mathcal{S}' \cup \{(p, p')\}, \Lambda') := \\
&\qquad \Big(\Big[\bigcup_{(s,c,r,s') \in P_{p \to p'}} \{c\} \cdot WM(R(s, s', \mathcal{S}', \Lambda' \cup \{\square_{pp'}\})) \cdot \{r\} \Big]^{\curvearrowleft \square_{pp'}} \Big) \curvearrowleft_{\square_{pp'}} \\
&\qquad \Big[\bigcup_{(s,c,r,s') \in P_{p \to p'}} \{c\} \cdot WM(R(s, s', \mathcal{S}', \Lambda')) \cdot \{r\} \Big]
\end{aligned}$$

Note that $S(p, p', \mathcal{S}' \cup \{(p, p')\}, \Lambda')$ represents the set of words $w \in MWM(\widetilde{\Sigma}_{\Lambda'})$ such that there is a summary of $\mathcal{P}_\Lambda$ over w from p to p' which uses only sub-summaries from

[2] note that if $w \in WM(\widetilde{\Sigma}_{\Lambda'})$, then $\beta = \bot$.

$\mathcal{S}' \cup \{(p,p')\}$. By the induction hypothesis, the sets $\mathit{WM}(R(s,s',\mathcal{S}',\Lambda' \cup \{\Box_{pp'}\}))$ and $\mathit{WM}(R(s,s',\mathcal{S}',\Lambda'))$ used in the definition of $S(p,p',\mathcal{S}' \cup \{(p,p')\},\Lambda')$ can be effectively denoted by VRE. Thus, one can construct a VRE over $\widetilde{\Sigma}_\Lambda$ denoting the language $S(p,p',\mathcal{S}' \cup \{(p,p')\},\Lambda')$. Now, we observe that:

$$\begin{aligned} \mathit{WM}(R(q,q',\mathcal{S}' \cup \{(p,p')\},\Lambda')) &= \mathit{WM}(R(q,q',\mathcal{S}',\Lambda')) \cup \\ &\quad \mathit{WM}(R(q,q',\mathcal{S}',\Lambda' \cup \{\Box_{pp'}\})) \curvearrowleft_{\Box_{pp'}} S(p,p',\mathcal{S}' \cup \{(p,p')\},\Lambda') \\ R(q,q',\mathcal{S}' \cup \{(p,p')\},\Lambda') &= R(q,q',\mathcal{S}',\Lambda') \cup \\ &\quad R(q,q',\mathcal{S}',\Lambda' \cup \{\Box_{pp'}\}) \curvearrowleft_{\Box_{pp'}} S(p,p',\mathcal{S}' \cup \{(p,p')\},\Lambda') \end{aligned}$$

Thus, by the induction hypothesis, it holds that the languages $\mathit{WM}(R(q,q',\mathcal{S}' \cup \{(p,p')\},\Lambda'))$ and $R(q,q',\mathcal{S}' \cup \{(p,p')\},\Lambda')$ can be effectively denoted by VRE. Moreover, by expanding recursively the above equalities until the base case (the number of iterations is at most $|Q|^2$), we deduce that each of the constructed VRE has size singly exponential in the size of the NVPA $\mathcal{P}$. This concludes the proof of the theorem. ◀

By Corollary 10 and Theorem 11, we obtain the following result.

▶ **Corollary 12.** *(Pure) Visibly Rational Expressions capture the class of VPL.*

4 Pure VRE

In this section, first, we show that pure VRE can be compositionally translated in *quadratic time* into equivalent NVPA. The key of the proposed efficient and elegant translation is represented by a subclass of NVPA, we call *strong* NVPA. Then, in Subsection 4.1, we establish the exact complexity of some language decision problems for pure VRE. Fix a pushdown alphabet $\widetilde{\Sigma}$. In the following, we use an additional special stack symbol $\widehat{\bot}$.

▶ **Definition 13.** *A* strong *NVPA over $\widetilde{\Sigma}$ is an NVPA $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ over $\widetilde{\Sigma}$ such that $\widehat{\bot} \in \Gamma$ and the following holds:*
Initial State Requirement*: $q_{in} \notin F$ and there are no transitions leading to q_{in}.*
Final State Requirement*: there are no transitions from accepting states.*
Push Requirement*: every push transition from the initial state q_{in} pushes onto the stack the special symbol $\widehat{\bot}$.*
Pop Requirement*: for all $q,p \in Q$ and $r \in \Sigma_{ret}$, $(q,r,\bot,p) \in \Delta$ iff $(q,r,\widehat{\bot},p) \in \Delta$.*
Well-formed (semantic) Requirement: *for all $w \in \mathcal{L}(\mathcal{P})$, every initialized accepting run of $\mathcal{P}$ over w leads to a configuration whose stack content is in $\{\widehat{\bot}\}^* \cdot \bot$.*
Note that the initial state requirement implies that $\varepsilon \notin \mathcal{L}(\mathcal{P})$.

The push requirement is used in particular to implement in an efficient way M-substitution and S-closure. Moreover, the pop requirement ensures that pop operations which pop the special stack symbol $\widehat{\bot}$ have the same effect as popping the empty stack (i.e., the stack containing just the special bottom symbol $\bot$). This requirement and the well-formed requirement are used in particular to implement in an efficient way concatenation and Kleene closure. In the following, we first show that strong NVPA are “efficiently" closed under union, concatenation, and Kleene closure.

▶ **Theorem 14.** *Let $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ and $\mathcal{P}' = \langle Q', q'_{in}, \Gamma', \Delta', F'\rangle$ be two strong NVPA over $\widetilde{\Sigma}$. Then, one can construct in linear time*

1. *a strong NVPA accepting $\mathcal{L}(\mathcal{P}) \cup \mathcal{L}(\mathcal{P}')$ (resp., $\mathcal{L}(\mathcal{P}) \cdot \mathcal{L}(\mathcal{P}')$) with $|Q| + |Q'| + 1$ states and $|\Gamma| + |\Gamma'| - 1$ stack symbols, and*

2. *a strong* NVPA *accepting* $[\mathcal{L}(\mathcal{P})]^* \setminus \{\varepsilon\}$ *with* $|Q|+1$ *states and* $|\Gamma|$ *stack symbols.*

Proof. We prove Condition 2 (Condition 1 is simpler). Let $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ be a strong NVPA over $\widetilde{\Sigma}$ an q'_{in} be a fresh control state. Define $\mathcal{P}' = \langle Q \cup \{q'_{in}\}, q'_{in}, \Gamma, \Delta', F\rangle$, where Δ' is obtained from Δ by adding new transitions as follows. First, for each transition $t \in \Delta$ leading to an accepting state, we add the transition obtained from t by replacing the target state of t with the initial state q_{in}. Let Δ_0 be the resulting set of transitions. Then, Δ' is obtained from Δ_0 by adding the following transitions: for each transition $t \in \Delta_0$ from the initial state q_{in}, we add the transition obtained from t by replacing the source state of t with the new initial state q'_{in}. Note that the construction is identical to the classical one used for regular languages. Now, we prove that $\mathcal{P}'$ is a *strong* NVPA accepting $[\mathcal{L}(\mathcal{P})]^* \setminus \{\varepsilon\}$. Hence, the result follows. Since $\mathcal{P}$ is a strong NVPA, by construction, it follows that $\mathcal{P}'$ satisfies the initial and final state requirements and the push and pop requirements of Definition 13. Thus, it remains to show that $\mathcal{L}(\mathcal{P}') = [\mathcal{L}(\mathcal{P})]^* \setminus \{\varepsilon\}$ and $\mathcal{P}'$ satisfies the well-formed requirement.

Here, we show that $\mathcal{L}(\mathcal{P}') \subseteq [\mathcal{L}(\mathcal{P})]^* \setminus \{\varepsilon\}$ and $\mathcal{P}'$ satisfies the well formed requirement. Let $w \in \mathcal{L}(\mathcal{P}')$ (note that $w \neq \varepsilon$ since $\mathcal{P}'$ satisfies the initial state requirement) and π be an initialized accepting run of $\mathcal{P}'$ over w of the form $(q'_{in}, \perp) \xrightarrow{w} (q_{acc}, \beta)$ for some stack content β and $q_{acc} \in F$. We need to show that $w \in [\mathcal{L}(\mathcal{P})]^* \setminus \{\varepsilon\}$ and $\beta \in \{\widehat{\perp}\}^* \cdot \perp$. By construction, w is of the form $w = w_1 \cdot \ldots \cdot w_n$ for some $n \geq 1$, such that $w_1, \ldots, w_n$ are non-empty and there are runs $\pi_1, \ldots, \pi_n$ of $\mathcal{P}$ over $w_1, \ldots, w_n$, respectively, of the form

$$\pi_1 = (q_{in}, \perp) \xrightarrow{w_1} (p_1, \beta_1), \;\; \pi_2 = (q_{in}, \beta_1) \xrightarrow{w_2} (p_2, \beta_2), \;\ldots\;, \\ \pi_n = (q_{in}, \beta_{n-1}) \xrightarrow{w_n} (p_n, \beta_n)$$

where $p_i \in F$ for each $1 \leq i \leq n$, and $\beta = \beta_n$. We show by induction on i that $w_i \in \mathcal{L}(\mathcal{P})$ and $\beta_i \in \{\widehat{\perp}\}^* \cdot \perp$ for all $1 \leq i \leq n$, hence the result follows. Since π_1 is an initialized accepting run of $\mathcal{P}$ over w_1 and $\mathcal{P}$ satisfies the well-formed requirement, the result for the base case holds. For the induction step, let us consider the run $\pi_i = (q_{in}, \beta_{i-1}) \xrightarrow{w_i} (p_i, \beta_i)$ with $i > 1$. By the induction hypothesis, $\beta_{i-1} \in \{\widehat{\perp}\}^* \cdot \perp$. Moreover, β_i is of the form $\beta_i = \beta'_i \cdot \{\widehat{\perp}\}^m \cdot \perp$ for some $m \geq 0$, where β'_i consists of the symbols pushed on the stack along π_i on reading the unmatched call positions of w_i. Since $\mathcal{P}$ satisfies the pop requirement, we easily deduce that there is also an *initialized accepting* run of $\mathcal{P}$ over w_i of the form $\pi_i = (q_{in}, \perp) \xrightarrow{w_i} (p_i, \beta'_i \cdot \perp)$. Since $\mathcal{P}$ satisfies the well-formed requirement, $\beta'_i \in \{\widehat{\perp}\}^*$. Hence, $\beta_i \in \{\widehat{\perp}\}^* \cdot \perp$, and we are done. This concludes the proof of the theorem. ◀

Next we show that strong NVPA are "efficiently" closed under M-substitution and S-closure. For this, we need the following preliminary result.

▶ **Lemma 15.** *Let* $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ *be a strong* NVPA *over* $\widetilde{\Sigma}$. *Then, one can construct in linear time a strong* NVPA *over* $\widetilde{\Sigma}$ *accepting* $MWM(\mathcal{L}(\mathcal{P}))$ *with* $|Q|$ *states and* $|\Gamma|+1$ *stack symbols.*

▶ **Theorem 16.** *Let* $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ *and* $\mathcal{P}' = \langle Q', q'_{in}, \Gamma', \Delta', F'\rangle$ *be two strong* NVPA *over* $\widetilde{\Sigma}$, *and* $\square \in \Sigma_{int}$. *Then, one can construct in linear time: (1) a strong* NVPA *accepting* $(\mathcal{L}(\mathcal{P}))^{\circlearrowleft_\square}$ *with* $|Q|$ *states and* $|Q|+|\Gamma|+1$ *stack symbols, and (2) a strong* NVPA *accepting* $\mathcal{L}(\mathcal{P}) \curvearrowleft_\square \mathcal{L}(\mathcal{P}')$ *with* $|Q|+|Q'|$ *states and* $|\Gamma|+|\Gamma'|+|Q|$ *stack symbols.*

Proof. Here, we focus on Condition (1). Given a strong NVPA $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ over $\widetilde{\Sigma}$ such that $\mathcal{L}(\mathcal{P}) \subseteq MWM(\widetilde{\Sigma})$, we construct a strong NVPA $\mathcal{P}'$ accepting $(\mathcal{L}(\mathcal{P}))^{\circlearrowleft_\square}$ with $|Q|$ states and $|Q|+|\Gamma|$ stack symbols. Hence, by Lemma 15, Condition (1) in the theorem follows. W.l.o.g. we assume that Q and Γ are disjoint and all the transitions from the initial state are push transitions. The NVPA $\mathcal{P}'$ is given by $\mathcal{P}' = \langle Q, q_{in}, \Gamma \cup Q, \Delta', F\rangle$, where Δ' is obtained from Δ by adding the following transitions:

- *New Push transitions*: for each internal transition $(q, \square, p) \in \Delta$ – note that $q \neq q_{in}$ – and for each push transition from the initial state of the form $(q_{in}, c, q', \widehat{\bot}) \in \Delta$, we add the new push transition (q, c, q', p).
- *New Pop transitions*: for each pop transition $(q, r, \widehat{\bot}, q_{acc}) \in \Delta$ which pops the special stack symbol $\widehat{\bot}$ and leads to an accepting state $q_{acc} \in F$, we add for each $p \in Q \setminus \{q_{in}\}$, the new pop transition (q, r, p, p).

Correctness of the construction directly follows from the following claim.

Claim: $\mathcal{P}'$ is a strong NVPA over $\widetilde{\Sigma}$ accepting $[\mathcal{L}(\mathcal{P})]^{\circlearrowleft_\square}$. ◀

Now, we can prove the main result of this section.

▶ **Theorem 17.** *Let E be a* pure *VRE. Then, one can construct in quadratic time an NVPA $\mathcal{P}$ accepting $\mathcal{L}(E)$ with at most $|E| + 1$ states and $|E|^2$ stack symbols.*

Proof. Since one can trivially check in linear time whether $\varepsilon \in \mathcal{L}(E)$, it suffices to show that one can construct in quadratic time a *strong* NVPA accepting $\mathcal{L}(E) \setminus \{\varepsilon\}$ with at most $|E| + 1$ states and $|E|^2$ stack symbols. The proof is by induction on $|E|$. The base case is trivial. For the induction step, the result easily follows from the induction hypothesis and Theorems 14 and 16. As example, we illustrate the case where $E = E_1 \curvearrowleft_\square E_2$. By the induction hypothesis, one can construct two strong NVPA $\mathcal{P}_1 = \langle Q_1, q^1_{in}, \Gamma_1, \Delta_1, F_1\rangle$ and $\mathcal{P}_2 = \langle Q_2, q^2_{in}, \Gamma_2, \Delta_2, F_2\rangle$ accepting $\mathcal{L}(E_1) \setminus \{\varepsilon\}$ and $\mathcal{L}(E_2) \setminus \{\varepsilon\}$, respectively. Moreover, $|Q_1| \leq |E_1|+1$, $|Q_2| \leq |E_2|+1$, $|\Gamma_1| \leq |E_1|^2$, and $|\Gamma_2| \leq |E_2|^2$. By Theorem 16, one can construct in linear time a strong NVPA $\mathcal{P} = \langle Q, q_{in}, \Gamma, \Delta, F\rangle$ accepting $(\mathcal{L}(E_1)\setminus\{\varepsilon\}) \curvearrowleft_\square (\mathcal{L}(E_2)\setminus\{\varepsilon\}) = \mathcal{L}(E) \setminus \{\varepsilon\}$. Moreover, $|Q| = |Q_1| + |Q_2|$ and $|\Gamma| = |\Gamma_1| + |\Gamma_2| + |Q_1|$. Hence, $|\Gamma| \leq |E_1|^2 + |E_2|^2 + |E_1| + 1 \leq (|E_1| + |E_2| + 1)^2 = |E|^2$ and $|Q| \leq |E_1| + |E_2| + 2 = |E| + 1$, and the result follows. ◀

4.1 Decision Problems for pure VRE

In this section, we show the following result.

▶ **Theorem 18.** *The universality, inclusion, and language equivalence problems for pure VRE are EXPTIME-complete.*

Sketched proof. The upper bounds directly follow from Theorem 17 and EXPTIME-completeness of universality, inclusion, and equivalence for NVPA [4]. For the lower bounds, it is sufficient to show EXPTIME-hardness for the universality problem. This is proved by a polynomial time reduction from the word problem for polynomial space bounded alternating Turing Machines (TM) with a binary branching degree, which is a well-known EXPTIME-complete problem [11]. Fix such a machine $\mathcal{M}$ with input alphabet A and set of states Q. Since $\mathcal{M}$ is polynomial space bounded, there is an integer constant $k \geq 1$ such that for each $\alpha \in A^*$, the space needed by $\mathcal{M}$ on the input α is bounded by $|\alpha|^k$. Fix an input α and let $n = |\alpha|$. W.l.o.g. we can assume that $k = 1$, $n > 1$, and each (reachable) TM configuration (from the fixed input α) can be described by a word in $A^* \cdot (Q \times A) \cdot A^*$ of length exactly n. Let T_{full} be the configuration-labeled binary tree corresponding to the unwinding of $\mathcal{M}$ from the initial configuration associated with the input α. A computation tree T of $\mathcal{M}$ (over α) is a *finite* tree obtained from T_{full} by pruning subtrees rooted at children of nodes labeled by existential configurations; T is accepting if each leaf is labeled by an accepting

TM configuration. $\mathcal{M}$ accepts α if there is an accepting computation tree (over α). We use a standard encoding of computation trees T by minimally well-matched words w_T over a suitable pushdown alphabet $\widetilde{\Sigma}$ [9, 4], where the given tree T is processed in depth-first order. This encoding ensures the following crucial property: for all nodes x and y of T labeled by TM configurations C_x and C_y such that y is the child of x, there is a subword of w_T encoding $C_x \cdot C_y$ or its reverse. Let $\mathit{Codes}(\alpha)$ be the set of words $w \in \mathit{MWM}(\widetilde{\Sigma})$ encoding accepting computation trees (over α). Then, we show that it is possible to construct in time polynomial in n and the size of $\mathcal{M}$ a pure VRE over $\widetilde{\Sigma}$ which denotes the language $\Sigma^* \setminus \mathit{Codes}(\alpha)$. Hence, the result follows. ◀

5 ω-Visibly Rational Expressions (ω-VRE)

In this section, we introduce the class of ω-Visibly Rational Expressions (ω-VRE) and provide a Büchi-like theorem for ω-VPL in terms of ω-VRE. Fix a pushdown alphabet $\widetilde{\Sigma}$. For a language $\mathcal{L}$ of finite words over Σ, we denote by $\mathcal{L}^\omega$ the standard ω-Kleene closure of $\mathcal{L}$.

▶ **Definition 19.** The syntax of ω-VRE I over $\widetilde{\Sigma}$ is inductively defined as follows:

$$I := (E)^\omega \;\big|\; (I \cup I) \;\big|\; (E \cdot I)$$

where E is a VRE over $\widetilde{\Sigma}$. Note that ω-VRE are defined similarly to ω-regular expressions. An ω-VRE I is *pure* if every VRE subexpression is pure. An ω-VRE I denotes a language of infinite words over Σ, written $\mathcal{L}(I)$, defined in the obvious way: $\mathcal{L}(E^\omega) = [\mathcal{L}(E)]^\omega$, $\mathcal{L}(I \cup I') = \mathcal{L}(I) \cup \mathcal{L}(I')$, and $\mathcal{L}(E \cdot I) = \mathcal{L}(E) \cdot \mathcal{L}(I)$.

We show that ω-VRE capture the class of ω-VPL. For this, we need the following preliminary result establishing that ω-VPL can be expressed in terms of VPL in the same way as ω-regular languages can be expressed in terms of regular languages.

▶ **Theorem 20.** *Let $\mathcal{L}$ be a ω-VPL with respect to $\widetilde{\Sigma}$. Then, there are $n \geq 1$ and VPL $\mathcal{L}_1, \mathcal{L}'_1, \ldots, \mathcal{L}_n, \mathcal{L}'_n$ with respect to $\widetilde{\Sigma}$ such that $\mathcal{L} = \bigcup_{i=1}^{i=n} \mathcal{L}_i \cdot (\mathcal{L}'_i)^\omega$. Moreover, the characterization is constructive.*

Since ω-VPL are effectively closed under ω-Kleene closure and under (left) concatenation with VPL (and the constructions can be done in linear time) [4], by Corollary 10, it follows that ω-VRE can be converted into equivalent ω-NVPA in single exponential time. Moreover, by using *strong* NVPA and constructions very similar to those used in the proof of Theorem 14, we can show that *pure* ω-VRE can be converted into equivalent Büchi ω-NVPA in quadratic time. Thus, by Theorem 20 we obtain the following result.

▶ **Theorem 21.** *(Pure) ω-VRE capture the class of ω-VPL. Moreover, pure ω-VRE can be converted in quadratic time into equivalent Büchi ω-NVPA.*

6 Conclusion

In this paper we have provided a Kleene/Büchi theorem for VPL. From a theoretical point of view, there are some interesting open questions. For example, the succinctness gap between VRE and NVPA (it is well-known that NFA are exponentially more succinct than regular expressions). From a practical point of view, it remains to be seen whether VRE are useful as a specification language for nested word search and for XML schemas. Another line of future work is the combination of VRE with temporal logics for nested words (like CaRet [2]), as done for word regular languages [13, 17].

References

1 R. Alur. Marrying words and trees. In *Proc. 26th PODS*, pages 233–242. ACM, 2007.

2 R. Alur, K. Etessami, and P. Madhusudan. A Temporal Logic of Nested Calls and Returns. In *Proc. 10th TACAS*, volume 2988 of LNCS, pages 467–481. Springer, 2004.

3 R. Alur, V. Kumar, P. Madhusudan, and M. Viswanathan. Congruences for Visibly Pushdown Languages. In *Proc. 32nd ICALP*, volume 3580 of LNCS, pages 1102–1114. Springer, 2005.

4 R. Alur and P. Madhusudan. Visibly Pushdown Languages. In *Proc. 36th STOC*, pages 202–211. ACM, 2004.

5 R. Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM*, 56(3), 2009.

6 M. Arenas, P. Barceló, and L. Libkin. Regular Languages of Nested Words: Fixed Points, Automata, and Synchronization. In *Proc. 34th ICALP*, volume 4596 of LNCS, pages 888–900. Springer, 2007.

7 T. Ball and S.K. Rajamani. Bebop: a symbolic model checker for boolean programs. In *Proc. 7th SPIN*, volume 1885 of LNCS, pages 113–130. Springer, 2000.

8 C. Bolduc and B. Ktari. Visibly Pushdown Kleene Algebra and Its Use in Interprocedural Analysis of (Mutually) Recursive Programs. In *Proc. 11th RelMiCS*, volume 5827 of LNCS, pages 44-58. Springer, 2009.

9 A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model-checking. In *Proc. 8th CONCUR*, volume 1243 of LNCS, pages 135–150. Springer, 1997.

10 L. Bozzelli. Alternating automata and a temporal fixpoint calculus for visibly pushdown languages. In *Proc. 18th CONCUR*, volume 4703 of LNCS, pages 476–491. Springer, 2007.

11 A.K. Chandra, D. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

12 A. Cyriac. Temporal Logics for Concurrent Recursive Programs. Rapport de Master, Master Parisien de Recherche en Informatique, Paris, France, September 2010.

13 D. Fisman, C. Eisner, and J. Havlicek. Formal syntax and Semantics of PSL: Appendix B of Accellera Property Language Reference Manual, Version 1.1. March, 2004.

14 J.E. Hopcroft and J.D. Ullman. *Introduction to automata theory, languages and computation.* Addison-Wesley, 1979.

15 S.C. Kleene. Representation of Events in Nerve Nets and Finite Automata. In *Automata Studies*, volume 34, pages 3–41. Princeton University Press, 1956.

16 V. Kumar, P. Madhusudan, and M. Viswanathan. Visibly pushdown automata for streaming XML. In *Proc. 16th WWW*, pages 1053–1062. ACM, 2007.

17 M. Leucker and C. Sánchez. Regular Linear Temporal Logic. In *Proc. 4th ICTAC*, volume 4711 of LNCS, pages 291–305. Springer, 2007.

18 C. Löding, P. Madhusudan, and O. Serre. Visibly Pushdown Games. In *Proc. 24th FSTTCS*, volume 3328 of LNCS, pages 408–420. Springer, 2004.

19 P. Madhusudan and M. Viswanathan. Query Automata for XML Nested Words. In *Proc. 34th MFCS*, volume 5734 of LNCS, pages 561–573. Springer, 2009.

20 C. Pitcher. Visibly Pushdown Expression Effects for XML Stream Processing. In *Proc. PLAN-X*, pages 1–14. ACM, 2005.

21 A. Thomo, S. Venkatesh, and Y.Y. Ye. Visibly Pushdown Transducers for Approximate Validation of Streaming XML. In *Proc. 5th FoIKS*, volume 4932 of LNCS, pages 219-238. Springer, 2008.

22 K. Thompson. Regular expression search algorithm. *Comm. of the ACM*, 11(6):419–422, 1968.