On Practical Regular Expressions  
(Preliminary Report)

Holger Petersen  
Reinsburgstr. 75  
70197 Stuttgart  
Germany  
August 7, 2014

Abstract

We report on simulation, hierarchy, and decidability results for Practical Regular Expressions (PRE), which may include back references in addition to the standard operations union, concatenation, and star.

The following results are obtained:

• PRE can be simulated by the classical model of nondeterministic finite automata with sensing one-way heads. The number of heads depends on the number of different variables in the expressions.

• A space bound $O(n \log m)$ for matching a text of length $m$ with a PRE with $n$ variables based on the previous simulation. This improves the bound $O(nm)$ from (Câmpeanu and Santean 2009).

• PRE cannot be simulated by deterministic finite automata with at most three sensing one-way heads or deterministic finite automata with any number of non-sensing one-way heads.

• PRE with a bounded number of occurrences of variables in any match can be simulated by nondeterministic finite automata with one-way heads.

• There is a tight hierarchy of PRE with a growing number of non-nested variables over a fixed alphabet. A previously known hierarchy was based on nested variables and growing alphabets (Larsen 1998).

• Greibach’s Theorem applies to languages characterized by PRE.

• The decidability of universality of PRE over a single letter alphabet is linked to the existence of Fermat Primes.
1 Introduction

Regular expressions have evolved from a tool for the analysis of nerve nets [12] into an important domain-specific language for describing patterns. The original set of operations (union, concatenation, and star) of what will be called “Classical Regular Expressions” (CRE) here has been enhanced in several different ways. A natural additional operation is complementation, leading to Extended Regular Expressions (ERE). On the language level complementation does not add power to regular expressions, since any regular expression can be converted into an equivalent finite finite automaton, which in turn can be complemented via the power-set construction. The matching problem however becomes harder [11, 15] (under the assumption that the corresponding complexity classes are different, see the table below) and an equivalence test certainly becomes infeasible (its memory requirement grows faster than any exponential function, [17, 6]). An operation that leads to a less complex equivalence problem is intersection (complete in exponential space, [9, 16]). Expressions based on the operations union, concatenation, star, and intersection are called Semi-Extended Regular Expressions (SERE).

Practical Regular Expressions (PRE) including several extensions have been implemented in operating system commands, data base query languages, and text editors. PRE include many syntactical enhancements like notations for sets of symbols (wildcards, ranges, enumerations of symbols etc.), optional subexpressions or bounded repetition of subexpressions. The latter operation has been investigated for the special case of squaring in [14], where completeness in exponential space was shown. An extension of PRE that goes beyond regular languages in expressive power is the use of back references. The $k$-th subexpression put into parentheses can be referenced by \textbackslash k, which matches the same string that is matched by the subexpression. This is the notation employed in many implementations, while the definition of [2] admits variable names. Also the exact semantics of the expressions vary. In [13] several syntactical criteria are imposed on PRE such that no variable can be used before it is defined. In contrast [3] allows for such a situation and assigns the empty set to the variable. We will adopt the latter definition here that will prevent a match with an uninstantiated variable.

As an example of a non-regular (and not even context-free) language consider the expression

$$\{(a(b))^*\}$$

characterizing the language $L_d = \{ww \mid w \in \{a, b\}^*\}$ of “double-words”. Many regular languages can be defined more succinctly by PRE than by CRE, like

$$(a|b)^* (a|b)(a|b)^* \backslash 2(a|b)^*$$

which characterizes all strings over \{a, b\} with (at least) one repeated oc-
The increase in expressive power of PRE has an impact on the complexity of decision problems. The matching problem for PRE is NP-complete due to back references [2] and equivalence is even undecidable [5].

Some results and references are summarized in the following table:

|                | PRE                  | ERE                  | CRE                  |
|----------------|----------------------|----------------------|----------------------|
| matching,      | NP-complete          | P-complete           | NL-complete          |
| member [2 Thm.|                      | [13 Thm.1,          |                      |
| 6.2]           |                      | [11 Thm. 2.2]       |                      |
| equivalence    | undecidable          | NSPACE $2^{\cdot g(n)}$ | PSPACE-compl.        |
|                |                      | $g(n) = n^{17}$ (u.b.) |                      |
|                |                      | $g(n) = c \cdot n(\log^* n)^2$ (l.b.) |                  |
|                | $\in ALOGTIME$      | see equivalence      | $\in ALOGTIME$       |
|                | (see CRE )          |                      |                      |

2 Simulation Results

**Theorem 1** Every PRE with $k$ different variables can be simulated by a nondeterministic finite automaton with $2k + 2$ sensing one-way heads.

**Proof.** The simulating NFA $M$ stores the PRE $\alpha$ of length $n$ in its finite control with one of the $n + 1$ positions marked (initially the position before the PRE). The heads form pairs $\ell_i, r_i$ for $1 \leq i \leq k + 1$, where pair $i \leq k$ corresponds to variable $v_i$. All heads move in parallel along the input string while $M$ parses $\alpha$ moving the marked position until an opening parenthesis is encountered. In this case head $\ell_1$ remains stationary while the other heads advance. When the matching closing parenthesis is encountered, head $r_1$ remains at the current position. In this way the value of $v_1$ is stored and similarly for each of the variables a sub-string of the input is marked. If $v_i$ occurs in $\alpha$, $M$ leaves head $\ell_{k+1}$ at the current position and advances $\ell_i$ comparing the symbols read with the input. Notice that at least $r_{k+1}$ is available for the comparison. When $\ell_i$ and $r_i$ meet, the value of $v_i$ has been compared to the input and a copy of $v_i$ is marked by $\ell_{k+1}$ and $r_{k+1}$. Now $M$ advances $\ell_i$ to the position of $\ell_{k+1}$ and then $\ell_{k+1}$ and $r_i$ to the position of $r_{k+1}$. In this way the value of $v_i$ is again marked.

If in this way $M$ is able to scan all of its input, it accepts. If a mismatch is detected, $M$ rejects. □
Theorem 2. The class of languages accepted by nondeterministic finite automata with sensing one-way heads properly includes those characterized by PRE.

Proof. Inclusion follows from Theorem 1. The language
\[
S = \{a^i b a^{i+1} b a^k \mid k = i(i+1)k' \text{ for some } k' > 0, i > 0\}
\]
is shown not to be generated by any PRE in [4]. A deterministic finite automaton with 3 sensing one-way heads can check divisibility of the length of the trailing block of length by \(i\) or \(i+1\) respectively (move heads with distance \(i\) or \(i+1\) over the block marking the last position with the third head). Since languages accepted by these finite automata are closed under intersection, the claim follows.

Finite multi-head automata can be simulated in nondeterministic logarithmic space. We obtain the following improvement of [3, Corollary 7]:

Corollary 1. The uniform membership problem for PRE has nondeterministic space complexity \(O(n \log m)\) where \(n\) is the number of pairs of parentheses in the PRE and \(m\) is the length of text.

It is natural to ask whether a simpler model of computation than nondeterministic finite automata with sensing heads can simulate PRE. The next results provide partial answers.

Theorem 3. Languages characterized by PRE with one variable cannot in general be accepted by deterministic finite automata with at most three sensing one-way heads or any number of non-sensing one-way heads.

Proof. Consider the language
\[
M = \{p\#t_1 pt_2 \mid p, t_1, t_2 \in \{0, 1\}^*\}
\]
formalizing the string-matching problem of deciding whether pattern \(p\) occurs in a given text \(t\). Notice that answers for more realistic problems like reporting the first or even every position of \(p\) in \(t\) would also solve the membership problem of \(M\).

A PRE specifying \(M\) is
\[
((0|1)*\#(0|1)*\setminus 1(0|1)*.
\]
By the result in [7] string-matching cannot be done with three sensing heads and by the result in [10] string-matching cannot be done with non-sensing one-way heads by deterministic finite automata. ☐
The above result shows that even very simple PRE cannot be simulated by deterministic finite automata with non-sensing one-way heads. This is not true for nondeterministic automata. We first define a notion of complexity of PRE depending on the number of variables occurring in a match of an expression. Let $c$ be the following function from PRE to $\mathbb{N} \cup \{\omega\}$ (where $\omega$ is greater than any element of $\mathbb{N}$):

- $c(n) = 1$ for $n \in \mathbb{N}$
- $c(a) = 0$ for $a \in \Sigma$
- $c(\alpha|\beta) = \max(c(\alpha), c(\beta))$
- $c(\alpha\beta) = c(\alpha) + c(\beta)$
- $c(\alpha^*) = \omega$ if $c(\alpha) \geq 1$
- $0$ if $c(\alpha) = 0$

**Theorem 4** Every PRE $\alpha$ with $k < \omega$ occurrences of variables ($c(\alpha) = k$) can be simulated by a nondeterministic finite automaton with $2^k + 1$ one-way heads.

**Proof.** One head of the simulator $A$ is used for matching the regular expression with the input. The definition of the $i$-th variable is marked on the input by a pair of heads for every occurrence of \( i \). When a \( i \) occurs, then the trailing head of the pair simulating this occurrence is moved along the input comparing the segments of the input until $A$ guesses coincidence of the two heads. Then $A$ moves both heads in parallel until at least one of them reaches the right end-marker. If they do not reach the end-marker at the same time, then $A$ rejects. \( \square \)

### 3 Hierarchy Results

In [13] languages defined by PRE with a growing number of variables are investigated. The complexity of these languages depends on two features of the construction:

- The cardinality of alphabets increases with the number of variables.
- Variables are nested.

Using the simulation from the proof of Theorem 2 we can establish:

**Theorem 5** The class of languages characterized by PRE forms an infinite hierarchy with respect to the number of variables, even when restricting PRE to a fixed alphabet and non-nested variables.
Proof. It is immediate from the definition that languages characterized by PRE with \( k \) variables form a subset of those characterized by PRE with \( k' \geq k \) variables.

The language

\[
L_b = \{w_1\#w_2\#\cdots\#w_b\#w_b\#\cdots\#w_2\#w_1 \mid \forall 1 \leq i \leq b : w_i \in \{0, 1\}^*\}
\]

can be defined by

\[
((0|1)\#((0|1)\#\cdots((0|1)\#\ \#2b - 1\#\ \#2b - 3\cdots\#3\#\})
\]

with \( b \) subexpression \(((0|1)\#)\). In [18] it is shown that nondeterministic finite automata with \( h \) one-way heads cannot accept \( L_b \) for \( b > \binom{h}{2} \). The main result of [18] also holds for sensing heads, see the remark on p. 337.

Every language characterized by a PRE with \( k \) variables can be accepted by a nondeterministic finite automaton with \( h = 2k + 2 \) sensing one-way heads according to Theorem \( \Box \). Let \( b = \binom{h}{2} + 1 = \binom{2k + 2}{2} + 1 \). Now \( L_b \) is a language characterized by a PRE with \( b \) variables that cannot be characterized with the help of at most \( k \) variables. \( \Box \)

We now improve the coarse separation of the previous hierarchy result using the concept of Kolmogorov complexity. The argument is again based on the languages \( L_b \), but we will show that \( L_b \) requires \( b \) variables.

**Theorem 6** The class of languages characterized by PRE with \( b \) non-nested variables properly includes those characterized by PRE with \( b - 1 \) variables for every \( b > 0 \).

First we prove the following Non-Matching Lemma for an input \( x \in L_b \) that is based on an incompressible string \( w \). Let \( w \in \{0, 1\}^* \) be an incompressible string of length \( n \) sufficient large and resulting in integer valued formulas in the following definitions. Form a word

\[
x = w_1\#w_2\#\cdots\#w_b\#w_b\#\cdots\#w_2\#w_1 \in L_b
\]

with \( |w_i| = n/b \) for all \( 1 \leq i \leq b \).

**Lemma 1 (Non-Matching Lemma)** Suppose that variable \( v \) is instantiated to a substring of length at least \( n/b + 2 + 18 \log n \) when matching string \( x \in L_b \) as defined above. Then \( v \) cannot match any other substring of \( x \).

**Proof.** For a contradiction suppose that \( v \) matches another substring of \( x \). We will identify \( v \) and the value assigned to it in the following discussion. Since \( v \) includes at least two symbols \#, the structure of \( v \) is \( w'\#w_i\#w'' \) for some \( 1 \leq i \leq b \) and \( w', w'' \in \{0, 1, \#\}^* \). We will show how to compress \( w \) by copying a substring of \( w \) from position \( p_1 \) through \( p_2 \) to a position \( p_3 \) in \( x \).
The first case we consider is that the portion \(#w_i\#\) of \(v\) matches a substring \(#w_j\#\) for \(j \neq i\) (in the left or right half of \(x\)). Then we can take \(p_1\) and \(p_2\) as the positions of the first and the last symbol of \(w_i\) and \(p_3\) as the position of the first symbol of \(w_j\).

The other case is that \(v\) matches the corresponding \(w'\#w_i\#w''\) from the other half of \(x\). By the length condition on \(v\), at least one of \(|w'|, |w''|\) is not less than \(9 \log n\). Suppose \(|w'| \geq 9 \log n\) and \(i = b\). Then \(w'\) is a suffix of \(w_1\#w_2\#\cdots\#w_b\) as well as of \(w_1\#w_2\#\cdots\#w_{b-1}\) and we can take \(p_1 = n - n/b - 9 \log n\), \(p_2 = n - n/b\), and \(p_3 = n - 9 \log n\). If \(i < b\) then \(w'\) is a suffix of \(w_1\#w_2\#\cdots\#w_{i-1}\) as well as of \(w_1\#w_2\#\cdots\#w_i\#w_{i+1}\). We take \(p_1 = (i - 1)n/b - 9 \log n\), \(p_2 = (i - 1)n/b\), and \(p_3 = (i + 1)n/b - 9 \log n\). Now consider the case \(|w''| \geq 9 \log n\). If \(i = b\) then \(w''\) is a prefix of \(w_b\#\cdots\#w_2\#w_1\) and of \(w_{b-1}\#\cdots\#w_2\#w_1\). We take \(p_1 = (b - 2)n/b + 1\), \(p_2 = (b - 2)n/b + 9 \log n\) and \(p_3 = (b - 1)n/b + 1\). If \(i < b\) then \(w''\) is a prefix of \(w_{i+1}\#\cdots\#w_2\#w_1\) as well as of \(w_{i-1}\#\cdots\#w_2\#w_1\). We take \(p_1 = (i - 2)n/b + 1\), \(p_2 = (i - 2)n/b + 9 \log n\) and \(p_3 = in/b + 1\).

The string \(w\) can be reconstructed from the following information:

- A formalization of this description including the recovery algorithm below \((O(1)\) bits).
- The values of \(n\), \(p_1\), \(p_2\), and \(p_3\) in self-delimiting binary form \((8 \log n\) bits).
- \(w_1w_2\cdots w_b\) with the portion of length \(p_2 - p_1 + 1\) starting at position \(p_3\) deleted (at most \(n - 9 \log n\) bits).

The string \(w = w_1w_2\cdots w_b\) can be reconstructed by copying \(p_2 - p_1 + 1\) symbols starting at position \(p_1\) inserting them at position \(p_3\). For \(n\) sufficiently large a compression by \(\log n - O(1)\) bits is obtained, contradicting the choice of \(w\).

We now continue the proof of Theorem 5. Let \(\alpha\) be a PRE with at most \(b - 1\) variables characterizing \(L_b\). Fix a matching of \(\alpha\) and \(x\) by recording the corresponding alphabet symbols of \(\alpha\) and \(x\). The string \(w\) can be reconstructed from the following information:

- A formalization of this description including the recovery algorithm below \((O(1)\) bits).
- The value of \(n\) \((O(\log n)\) bits).
- PRE \(\alpha\) \((O(1)\) bits).
- The occurrence of \('#'\) in \(\alpha\) matching the center of \(x\) \((O(1)\) bits).
- For every variable \(v_i\) its state when the center has just been matched:
1. \( v_i \) is instantiated and \( |v_i| \geq n/b + 2 + 18 \log n \).
2. \( v_i \) is instantiated and \( |v_i| < n/b + 2 + 18 \log n \).
3. \( v_i \) is partially instantiated and \( |v_i| \geq n/b + 2 + 18 \log n \).
4. \( v_i \) is partially instantiated and \( |v_i| < n/b + 2 + 18 \log n \).
5. \( v_i \) has not been instantiated.

(\( O(1) \) bits).

- For every (partially) instantiated variable with \( |v_i| < n/b + 2 + 18 \log n \) its current value when the center has just been matched ((\( b - 1 \))\( n/b + 2 + 18 \log n \) bits).

For recovering the string \( w \) try to determine a matching of \( \alpha \) and every \( w_3\# \cdots w_2\# w_1 \) with \( w_i \in \{0, 1\}^{n/b} \) by backtracking. The matching is based on the recorded values of the variables, where partially instantiated variables are extended until the corresponding closing parenthesis is encountered. The matching starts at the recorded occurrence of \( \# \) in \( \alpha \). If a variable \( v \) is encountered in \( \alpha \) and \( v \) is of type 1, 3, 4, or 5 and in the latter two cases the current matching extends the value to a length at least \( n/b + 2 + 18 \log n \), then the matching fails due to the Non-Matching Lemma. If a variable occurs that has not been instantiated, the matching fails as well. If for \( w_3\# \cdots w_2\# w_1 \) a matching has been determined, the string \( w = w_1 w_2 \cdots w_b \) has been reconstructed.

The description of \( w \) has length \( (b - 1)n/b + O(\log n) = n - n/b + O(\log n) \) leading to a compression for \( n \) sufficiently large and thus contradicting the choice of \( w \).

\( \square \)

4 Decidability Results

Greibach’s Theorem \( \square \) states that any nontrivial property \( P \) of a class \( C \) of formal languages over an alphabet \( \Sigma \cup \{\#\} \) is undecidable, provided that the following conditions are satisfied:

1. The languages in \( C \) have finite descriptions.
2. \( C \) contains every regular language over \( \Sigma \cup \{\#\} \).
3. For descriptions of languages \( L_1, L_2 \in C \) and regular language \( R \in C \), descriptions of \( L_1 R, RL_1, \) and \( L_1 \cup L_2 \) can be computed effectively.
4. Universality (\( L = \Sigma^*? \)) is undecidable for \( L \in C \) with \( L \subseteq \Sigma^* \).
5. \( P \) is closed under quotient by each symbol in \( \Sigma \cup \{\#\} \).
Let \( C \) be the class of languages characterized by \( \text{PRE} \). The finite descriptions are \( \text{PRE} \) (Property\( ^1 \)), regular languages are characterized by \( \text{PRE} \) without variables (Property\( ^2 \)), \( \text{PRE} \) can be composed (Property\( ^3 \)), Freydenberger\( ^5 \) has shown universality to be undecidable (Property\( ^4 \)). Freydenberger’s results of undecidability of regularity and cofiniteness follow from Greibach’s Theorem. We can extend this list by context-freeness.

For \( \text{PRE} \) over a single letter alphabet we do not have an undecidability result for equivalence, but we can provide evidence that a decision procedure will not be obvious by linking it to Fermat Primes\( ^1 \) Sequence A019434].

**Theorem 7** If for \( \text{PRE} \) over a single letter alphabet equivalence is effectively decidable, we can solve the open problem whether 3, 5, 17, 257 and 65537 are the only prime Fermat numbers.

**Proof.** The \( \text{PRE} \) \( \alpha_1 \) describes all strings of length greater than two except those with a length of the form \( 2^n + 1 \) for \( n \geq 1 \):

\[
\alpha_1 = ( (aa) + a ) \ast 1 \ast a .
\]

Expression \( \alpha_1 \) describes all nonempty strings having a length with a proper divisor, thus omitting all primes and one :

\[
\alpha_2 = (a + a) \ast 1 + .
\]

We combine the expressions \( \alpha_1 \) and \( \alpha_2 \) with strings of length 3, 5, 17, 257 and 65537 (the known Fermat primes) and add 0, 1, and 2:

\[
\alpha_3 = ( (aa) + a ) \ast a \ast (a + a) \ast 1 + |a^0|a^1|a^3|a^5|a^{17}|a^{257}|a^{65537}.
\]

Now \( \alpha_3 \) is equivalent to \( a^* \) if and only if no additional Fermat prime exists. \( \Box \)

### 5 Discussion

We have established connections between \( \text{PRE} \) and classical models of computation within \( \text{NSPAC} (\log n) \). The hierarchy of \( \text{PRE} \) with a growing number of variables has been strengthened to expressions without nested variables and a fixed alphabet. We conjecture that equivalence for \( \text{PRE} \) over a single letter alphabet is undecidable.

**References**

[1] The On-Line Encyclopedia of Integer Sequences. https://oeis.org/ (download August 5, 2014).
[2] A. V. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume A, Algorithms and Complexity*, pages 255–300. MIT Press, Cambridge, MA, 1990.

[3] C. Câmpeanu and N. Santean. On the intersection of regex languages with regular languages. *TCS*, 410:2336–2344, 2009.

[4] B. Carle and P. Narendran. On extended regular expressions. In *Proc. LATA 2009*, number 5457 in Lecture Notes in Computer Science, pages 279–289. Springer, 2009.

[5] D. D. Freydenberger. Extended regular expressions: Succinctness and decidability. In T. Schwentick and C. Dürr, editors, *Proceedings of the 28th Annual Symposium on Theoretical Aspects of Computer Science (STACS11)*, Leibniz International Proceedings in Informatics, pages 507–518, Schloss Dagstuhl, 2011. Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany.

[6] M. Fürrer. *Nicht-elementare untere Schranken in der Automaten-Theorie*. PhD thesis, ETH Zürich, 1978.

[7] M. Geréb-Graus and M. Li. Three one-way heads cannot do string matching. *Journal of Computer and System Sciences*, 48:1–8, 1994.

[8] S. Greibach. A note on undecidable properties of formal languages. *Mathematical Systems Theory*, 2:1–6, 1968.

[9] H. B. Hunt III. The equivalence problem for regular expressions with intersection is not polynomial in tape. Report TR 73-161, Department of Computer Science, Cornell University, 1973.

[10] T. Jiang and M. Li. $k$ one-way heads cannot do string-matching. *Journal of Computer and System Sciences*, 53:513–524, 1996.

[11] T. Jiang and B. Ravikumar. A note on the space complexity of some decision problems for finite automata. *Information Processing Letters*, 40:25–31, 1991.

[12] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[13] K. S. Larsen. Regular expressions with nested levels of back referencing form a hierarchy. *Information Processing Letters*, 65:169–172, 1998.

[14] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of
the 13th Annual IEEE Symposium on Switching and Automata Theory, College Park (Maryland), pages 125–129, 1972.

[15] H. Petersen. Decision problems for generalized regular expressions. In Proceedings of the 2nd International Workshop on Descriptive Complexity of Automata, Grammars and Related Structures, London (Ontario), pages 22–29, 2000.

[16] J. M. Robson. The emptiness of complement problem for semi extended regular expressions requires $c^n$ space. Information Processing Letters, 9:220–222, 1979.

[17] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In Proceedings of the 5th ACM Symposium on Theory of Computing (STOC’73), Austin (Texas), pages 1–9, 1973.

[18] A. C. Yao and R. L. Rivest. $k + 1$ heads are better than $k$. Journal of the Association for Computing Machinery, 25:337–340, 1978.