On extended regular expressions

Benjamin Carle, Paliath Narendran, Colin Scheriff

On extended regular expressions

(Extended abstract)

Benjamin Carle, Paliath Narendran and Colin Scheriff
Dept. of Computer Science
University at Albany–SUNY
Albany, NY 12222

June 19, 2007

Abstract

In this paper we extend the work of Campeanu, Salomaa and Yu [3] on extended regular expressions featured in the Unix utility *egrep* and the popular scripting language *Perl.* We settle the open issue of closure under intersection and provide an improved pumping lemma that will show that a larger class of languages is not recognizable by extended regular expressions.

Introduction

Grep, a well-known command line search utility, is used regularly on Unix and other operating systems to find matching lines in files or standard input. Grep uses regular expressions to match patterns, thereby allowing a user to quickly find important data in very large files or command output. *Egrep,* a variant of grep, uses extended regular expressions to increase the set of languages recognizable by the utility. Because the set of languages recognized by egrep is larger than that of theoretical regular expressions, it is important to understand the expressive power of this utility.

In this paper we extend the pioneering work of [3]; we show that the family of languages recognizable by extended regular expressions is not closed under intersection, thereby settling an open problem. Furthermore, we introduce an improved pumping lemma and use that lemma to show a class of languages that satisfies the pumping property of [3] but not expressible by extended regular expressions.

Definitions

The syntax of extended regular expressions as in *egrep* and *Perl* is defined in [3]. Standard regular expressions, as specified in formal language theory, is extended using *backreferences.* The backreference $\backslash n$ stands for the string previously matched by the regular expression between the n^{th} left parenthesis and the corresponding right parenthesis. As is well-known, this significantly increases expressive power; for instance, the expression `((aa)+a)\1*` specifies the language

$$\{\, a^i \mid i > 0 \text{ and } i \text{ is not a power of } 2\}$$

which is not even context-free. Similarly `(a+)(b+)\1\2` specifies

$$\{\, a^i b^j a^i b^j \mid i, j > 0\}$$

which is not context-free either.

For clarity, let us number left parentheses, starting with 1, from the left. Give the same numbers to the corresponding (matching) right parentheses.

$$\underset{1}{(}\ldots\underset{2}{(}\ldots\underset{2}{)}\ldots\underset{3}{(}\ldots\underset{3}{)}\ldots\underset{1}{)}$$

As in [3] we assume that any occurrence of a backreference `\m` in an extended regular expression is preceded by $\underset{m}{)}$.

Matching a string with an extended regular expression (eregexp-matching) is often defined as follows (paraphrasing [1]):

1. If a is a symbol in the alphabet, then a matches a.
2. if r matches a string x, then $\underset{i}{(}\, r\, \underset{i}{)}$ matches x and the value x is assigned to $\backslash i$.
3. $\backslash j$ matches the string that has been assigned to it.
4. if r_1 and r_2 are regexps, then $r_1 \cup r_2$ matches any string matched by either r_1 or r_2.
5. if r_1 and r_2 are regexps, then $r_1 r_2$ matches any string of the form xy where r_1 matches x and r_2 matches y.
6. if r is a regexp, then r^* matches any string of the form $x_1 \ldots x_n$, $n \geq 0$, where r matches each x_i $(1 \leq i \leq n)$.

A more precise definition of a match is given in [3] using ordered trees. We give below that definition too, with a slight modification. Positions in an ordered tree are denoted by sequences of positive integers, with the empty sequence denoting the root position. (See [5] for a formal definition.) Note that left-to-right lexicographic order $\prec_{lex}$ among positions corresponds to *pre-order* traversal.

An ordered tree T is a valid match-tree for w and α if and only if:

1. The root of T has the label $(w,\ \alpha)$.
2. For every node $u \in dom(T)$,
   (a) if $T(u) = (w,\ a)$ for some $a \in \Sigma$, then u is a leaf node and $w = a$.

(b) if $T(u) = (w,\ \beta_1\beta_2)$, then u has two children labeled, respectively, by $(w_1,\ \beta_1)$ and $(w_2,\ \beta_2)$ where $w_1 w_2 = w$.

(c) if $T(u) = (w,\ \beta_1|\beta_2)$, then u has one child labeled by either $(w,\ \beta_1)$ or $(w,\ \beta_2)$.

(d) if $T(u) = (w,\ \beta^*)$, then either u is a leaf node and $w = \lambda$ or u has $k \geq 1$ children labeled by $(w_1,\ \beta), \ldots, (w_k,\ \beta)$ where each $w_i \in \Sigma^+$, and $w = w_1 \ldots w_k$.

(e) if $T(u) = (w,\ \underset{i}{(}\gamma\underset{i}{)})$, then it has one child labeled by $(w,\ \gamma)$.

(f) if $T(u) = (w,\ \backslash m)$, then u is a leaf node, $\underset{m}{(}\ \beta\ \underset{m}{)}$ is a subexpression of α, and there is a node v to the left of u such that $T(v) = (w,\ \underset{m}{(}\ \beta\ \underset{m}{)})$ and no node between v and u has $\underset{m}{(}\ \beta\ \underset{m}{)}$ in its label. In other words, w is the string previously (in the left-to-right pre-order) matched by $\underset{m}{(}\ \beta\ \underset{m}{)}$.

The difference between this definition and the one in [3] is that unassigned backreferences are not set to the empty string λ as default in our definition. Thus there is no valid match-tree for b and `((aa)|\2b)`.

The language denoted by an extended regular expression α is defined as

$$\mathcal{L}(\alpha) = \{\, w \in \Sigma^* \mid (w,\ \alpha) \text{ is the label at the root of a valid match-tree } \}.$$

A language L is an eregexp language if and only if there is an extended regexp α such that $L = \mathcal{L}(\alpha)$.

The Results

Campeanu, Salomaa and Yu [3] proved the following pumping lemma for eregexp languages. To the best of our knowledge, this is the only pumping lemma of its kind.

Lemma 1 (The CSY Pumping Lemma) [3] *Let α be an extended regexp. Then there is a constant $N > 0$ such that if $w \in L(\alpha)$ and $|w| > N$, then there is a decomposition $w = x_0 y x_1 y \cdots y x_m$, for some $m \geq 1$, such that*

1. $|x_0 y| < N$,
2. $|y| \geq 1$, *and*
3. $x_0 y^j x_1 y^j \cdots y^j x_m \in \mathcal{L}(\alpha)$ *for all* $j > 0$.

Lemma 2 *The language*

$$\mathcal{S} \;=\; \{\, a^i b a^{i+1} b a^k \mid k = i(i+1)k' \text{ for some } k' > 0, i > 0 \,\}$$

is not an eregexp language.

Proof: Assume $\mathcal{S}$ is regular and let N be the constant given by the CSY pumping lemma. Consider $w = a^N ba^{N+1} ba^{N(N+1)}$. Then there is a decomposition $w = x_0 y x_1 y ... y x_m$ for some $m \geq 1$ and from the pumping lemma $y = a^p$ for some $p \geq 1$. Since $|x_0 y| < N$ there must be at least one occurrence of y in a^N. Assume there are $q \geq 1$ occurrences of y in a^N. By (3) from the CSY pumping lemma there must also be q occurrences of y in a^{N+1} as otherwise $x_0 y^2 x_1 y^2 ... y^2 x_m \notin \mathcal{S}$. Let r be the number of occurrences of y in $a^{N(N+1)}$ and note that $N(N+1) \geq rp \geq 0$. Now consider $x_0 y^2 x_1 y^2 ... y^2 x_m = a^{N+qp} ba^{N+1+qp} ba^{N(N+1)+rp} \in \mathcal{S}$. Then

$$k_2(N+qp)(N+1+qp) = N(N+1) + rp$$

for some k_2. Since $rp \leq N(N+1)$, $N(N+1)+rp \leq 2(N(N+1))$. Since $qp \geq 1$, $(N+qp)(N+1+qp) \geq (N+1)(N+2) > N(N+1)$. Thus $k_2(N+qp)(N+1+qp) \geq k_2(N+1)(N+2) > k_2 N(N+1)$. $k_2 N(N+1) < 2(N(N+1))$ is only true for $k_2 = 1$. Thus we have $(N+qp)(N+1+qp) = N(N+1)+rp$, so

$$rp \quad = \quad q^2p^2 + qp(2N+1) \tag{1}$$

Now consider $x_0 y^3 x_1 y^3 ... y^3 x_m = a^{N+2qp} ba^{N+1+2qp} ba^{N(N+1)+2rp} \in \mathcal{S}$. Then it must be that

$$k_3(N+2qp)(N+1+2qp) = N(N+1) + 2rp$$

for some k_3. But note that $(N+2qp)(N+1+2qp) = N(N+1) + \mathbf{4q^2p^2} + 2qp(2N+1)$, whereas $N(N+1) + 2rp = N(N+1) + \mathbf{2q^2p^2} + 2qp(2N+1)$ by (1). Hence such a k_3 cannot exist. □

Lemma 3 *eregexp languages are not closed under intersection.*

Proof: The language $\mathcal{S}$ is the intersection of $\mathcal{L}($`(a+)b(\1a)b\1+`$)$ and $\mathcal{L}($`(a+)b(\1a)b\2+`$)$. □

We can show, by a reduction from the membership problem for phrase structured grammars, that

Lemma 4 *The following problem is undecidable:*

Instance:	*Two eregexps α and β.*
Question:	*Is $\mathcal{L}(\alpha) \cap \mathcal{L}(\beta)$ empty?*

Lemma 5 *Let α be an eregexp. Then for any $k > 0$ there are positive integers $N(k)$ and m such that if $w \in \mathcal{L}(\alpha)$ and $|w| > N(k)$ then w has a decomposition $w = x_0 y x_1 y \cdots y x_{m'}$ ($m' < m$) such that*

1. *$|y| \geq k$, and*
2. *$x_0 y^j x_1 y^j \cdots y^j x_{m'} \in \mathcal{L}(\alpha)$ for all $j > 0$.*

Proof: Let $N(k) = |\alpha| 2^t k$ where t is the number of backreferences in α. Then if $|w| > N$ there is a substring of w of length $\geq k$ that matches a Kleene star in α. (Each backreference can at most double the length of the word it matches.) Let $m = t + 1$.

Let $w = x_0 y z$ where y is the *rightmost* largest substring of w that matches a Kleene star. Then clearly $|y| \geq k$. Let t' equal the number of (direct or indirect) backreferences to any expression that contains this star. Let $m' = t' + 1$. Let $z = x_1 y x_2 y x_3 \ldots x_{m'}$ where the multiple instances of y correspond to these backreferences. Then $w = x_0 y x_1 y x_2 y \ldots y x_{m'}$ and clearly $x_0 y^j x_1 y^j x_2 y^j \ldots y^j x_{m'} \in \mathcal{L}(\alpha)$ for all $j \geq 1$. □

Lemma 6 *The language* $\{ wcw^R \mid w \in \{a, b\}^* \}$ *satisfies the CSY pumping property. (w^R stands for the reverse of w.)*

Lemma 7 *The language* $\{ wcw^R \mid w \in \{a, b\}^* \}$ *is not an eregexp language.*

Proof Sketch: Consider $x = (abaabb)^N$. It is not hard to see that x and x^R do not share any common substrings of length ≥ 5. Therefore, if we take $k = 5$, and the string $(abaabb)^{N(k)} c (bbaaba)^{N(k)}$, the pump occurs either to the left or the right of c, but not both. This leads to a contradiction when we pump. □

The matching problem for extended regular expressions has been shown to be NP-complete [1]. It turns out that the problem is NP-complete even if the target alphabet is unary:

Lemma 8 *The matching problem for extended regular expressions is NP-complete even when the target (subject) string is over a unary alphabet.*

Proof: Membership in NP follows from the earlier result. NP-hardness can be proved by a reduction from the vertex cover problem. □

References

[1] A.V. Aho. Algorithms for Finding Patterns in Strings. Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, 1990: 255-300.

[2] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences* 21 (1980) 46–62.

[3] C. Campeanu, K. Salomaa and S. Yu. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science* 14 (6) (2003) 1007–1018.

[4] G. Della Penna, B. Intrigila, E. Tronci and M. Venturini-Zili. Synchronized regular expressions. *Acta Informatica* 39 (1) (2003) 31–70.

[5] J.H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving.* Harper & Row, 1986.

[6] A. Hume. A tale of two greps. *Software–Practice and Experience* 18 (11) (1988) 1063–1072.