Regular Expressions with Nested Levels of Back Referencing Form a Hierarchy

Kim S. Larsen*
Odense University†

Abstract

For many years, regular expressions with back referencing have been used in a variety of software products in common use, including operating systems, editors, and programming languages. In these products, regular expressions have been extended with a naming construct. If a subexpression is named by some variable, whenever this subexpression matches some string, that string is assigned to the variable. Later occurrences of the same variable will then match the string assigned to it. This construction greatly increases the power of regular expressions and is useful in text searching as well as in text substitution in large documents. We study the nested usage of this operator, and prove that the power of the expressions increase with the number of nested levels that are allowed.

Keywords: Formal languages; regular expressions; back referencing.

1 Introduction

For many years, regular expressions with back referencing have been used in a variety of software products in common use, including operating systems, editors, and programming languages. However, it seems that a theoretical study of this language construction has not been made.

In [2], a vaguely related class of expressions are studied. In the terminology of our paper, these are regular expressions without alternation and with back referencing being restricted to using one variable name only.

Using the semantics of regular expressions with back referencing as outlined in [1], we define nested levels of back referencing and prove that the number of languages that can be expressed increases with the number of nested levels that are allowed.

*Supported in part by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT).

†Department of Mathematics and Computer Science, Odense University, Campusvej 55, DK–5230 Odense M, Denmark. Email: kslarsen@imada.ou.dk.

2 Regular expressions with back referencing

We choose a standard syntax for regular expressions with and without back referencing [1]. Let Σ, the *alphabet*, be an infinite[1] set of *symbols*, and let Ψ be an infinite set of *names*. The following grammar defines the syntax of regular expressions with back referencing.

$$E \ ::= \ a \ \mid \ E \cdot E \ \mid \ E|E \ \mid \ E^* \ \mid \ E\%\alpha \ \mid \ \alpha \ \mid \ (E)$$

where $a \in \Sigma$ and $\alpha \in \Psi$.

The precedence of the operators from highest to lowest is as follows: naming ($E\%\alpha$), Kleene star (E^*), concatenation ($E \cdot E$), and alternation ($E|E$). As usual, we often write EE instead of $E \cdot E$, i.e., leaving out the "dot". If E is a regular expression with back referencing, we use $L(E)$ to denote the language it defines.

A complete definition of the semantics of regular expressions with back referencing can be found in [1]. The semantics of the operators concatenation, alternation, and star are as usual for regular expressions. Informally, the semantics of naming and use of variables is best explained from an operational point of view. If we try to "parse" a string, from left to right, according to a regular expression with back referencing, whenever a substring is matched with a named subexpression, that substring is assigned to the variable in question. A later occurrence of this variable matches the string assigned to it.

Before continuing, let us give a few examples (partly from [1]) of the semantics of these expressions:

- $(a|b)^*\%\alpha \cdot \alpha \cdot \alpha$ is the language $\{www \mid w \in (a|b)^*\}$.
- $(a|b|c)^* \cdot (a|b|c)\%\alpha \cdot (a|b|c)^* \cdot \alpha \cdot (a|b|c)^*$ is the language of strings over $\{a, b, c\}$ with at least one repeated occurrence of a symbol.
- $(a|b)^*\%\alpha \cdot (a|b)^*\%\beta \cdot (\alpha|\beta)$ is the language $\bigcup_{u,v \in (a|b)^*} \{uuv, uvv\}$.
- $((a|b)^*\%\alpha \cdot \alpha)^*$ is the set of all strings over $\{a, b\}$ of the form $s_1 s_1 s_2 s_2 \cdots s_k s_k$ for some k, where the s_i's can be any strings of a's and b's.

In the last example, notice that in each iteration of the outermost star expression, a new value is bound to the variable α.

For convenience, we use e^i as an abbreviation for $e \cdot e \cdot \cdots$ (i times) $\cdot e$, and we assume that names are not reused, i.e., in any expression e, for any $\alpha \in \Psi$, there is at most one subexpression of the form $e'\%\alpha$. This does not restrict the expressive power and could be avoided at the cost of more cumbersome definitions and proofs.

[1] Usually, the alphabet for regular expressions is finite. Here, our goal is to establish an infinite hierarchy, and we need infinitely many symbols. However, each expression that we consider is finite and uses only a finite part of the alphabet. For purists, we want to remark that we could instead have used an infinite family of finite alphabets.

There are the following additional requirements, which are necessary in order to ensure that the use of variables is well-defined:

- If α appears in an expression e, then there must exist a subexpression e' of e of the form $e' = e_1 \cdot e_2$, where the α under discussion is a subexpression of e_2, and an expression of the form $e''\%\alpha$ is a subexpression of e_1.
- Assume that $e''\%\alpha$ is a subexpression of e. If there exists a subexpression $e' = e_1|e_2$ of e such that e'' is a subexpression of e_i, where $i = 1$ or $i = 2$, then the name α is only allowed to appear in e_i.
- Assume that $e''\%\alpha$ is a subexpression of e. If there exists a subexpression $e' = {e_1}^*$ of e such that e'' is a subexpression of e_1, then α is only allowed to appear in e_1.

The first requirement ensures that a name is defined before it is used, and the last two requirements ensure that a string has been assigned to it.

The *nested level* of a regular expression with back referencing is defined recursively in the structure of the expression. Regular expressions have nested level zero. The nested level of $E\%\alpha$ is one more than the nested level of E. The nested level of a variable α is the nested level of its defining expression $E\%\alpha$. For all other operators, the nested level of the expression is the maximum of the nested levels of the arguments.

In the following sections, we use *contexts* [3]. These are merely expressions with a "hole" instead of a symbol. As an example, $C[] = (a^* \cdot [])^*$ is a context. If $C[]$ is a context, $C[E]$ is the expression $C[]$ with the "hole" replaced by E. In the example, $C[b|c]$ is the expression $(a^* \cdot (b|c))^*$.

We use $\sigma(E)$ (the *signature* of E) to denote the set of symbols from Σ occurring in an expression E.

3 The hierarchy

Let $\mathcal{L}_i$ be the set of languages which can be expressed by a regular expression with back referencing of at most i nested levels. Thus, $\mathcal{L}_0$ equals the set of regular languages. Clearly, $\mathcal{L}_i \subseteq \mathcal{L}_j$, if $i \leq j$. In this section, we prove that $\mathcal{L}_i \neq \mathcal{L}_j$ if $i \neq j$.

Assume that $\Sigma = \{a_0^l, a_0^m, a_0^r, a_1^l, a_1^m, a_1^r, a_2^l, a_2^m, a_2^r, \ldots\}$ and $\Psi = \{\alpha_0, \alpha_1, \alpha_2, \ldots\}$. The superscripts l, m, and r stand for *left*, *middle*, and *right*.

Define the sequence of expressions $\{x_i\}_{i\geq 0}$ as follows: $x_0 = (a_0^l a_0^m a_0^r)^*$ and for $i \geq 1$, $x_i = (a_i^l \cdot x_{i-1}\%\alpha_{i-1} \cdot a_i^m \cdot \alpha_{i-1} \cdot a_i^r)^*$.

Clearly, $L(x_i) \in \mathcal{L}_i$. In this section, we show that for all $i > 0$, $L(x_i) \notin \mathcal{L}_{i-1}$. Thus, the sequence of sets of languages forms a hierarchy.

If a regular expression with back referencing R defines the language $L(x_k)$ for some k, then we can prove that R must have a certain form. Basically, the

following lemma breaks up the expression R. Looking at R as a syntax tree, it says that there must exist a path from the root to a leaf on which there are $k+1$ stars with the property that the symbols a_i^d are introduced gradually going up this path (there could be more than $k+1$ stars total; the statement is merely that there exists $k+1$ stars with this property). More precisely, only symbols a_j^d with $j < i$ are present in the subexpression under the ith of these stars going from the the leaf and up.

Lemma 1 Let R be a regular expression with back referencing such that $L(R) = L(x_k)$. Then there exists expressions R_i and contexts C_i, $1 \leq i \leq k+1$, such that R can be written as follows:

$$R = C_{k+1}[R_k{}^*] \qquad R_i = C_i[R_{i-1}{}^*],\ i \in \{1, \ldots, k\} \qquad R_0 = C_0[a_0^m]$$

where $\sigma(R_0) = \{a_0^l, a_0^m, a_0^r\}$ and $\sigma(R_i) = \sigma(R_{i-1}) \cup \{a_i^l, a_i^m, a_i^r\},\ i \in \{1, \ldots, k\}$.

Proof Let y_j be as x_k, except that all occurrences of stars in x_k are replaced by j's, i.e., any subexpression e^* is replaced by e^j. Let $M = \cup_j L(y_j)$. Clearly, $M \subseteq L(x_k)$ and M is infinite.

We consider all possible R_i's and C_i's and show that if they are not of the right form, then either strings not in $L(x_k)$ can be produced (which is a contradiction), or this part of the expression R can only contribute with finitely many of the substrings in M. This also leads to a contradiction since there are only a finite number of possibilities for choosing the R_i's and C_i's (which is the same as saying that there are only a finite number of different paths from the root to a leaf in the syntax tree of a finite expression).

For convenience, define $R_{k+1} = R$. Now, assume for the sake of contradiction that the lemma does not hold. Let $p \in \{0, \ldots, k\}$ be the largest integer for which we can find expressions and contexts of the right form, i.e., for $1 \leq i \leq p$, we have $R_i = C_i[R_{i-1}{}^*]$ and $R_0 = C_0[a_0^m]$, where $\sigma(R_0) = \{a_0^l, a_0^m, a_0^r\}$ and $\sigma(R_i) = \sigma(R_{i-1}) \cup \{a_i^l, a_i^m, a_i^r\}$. We consider the different possibilities for why this sequence cannot be extended to one fulfilling the lemma.

Consider the smallest context $C[]$ such that $R = C'[(C[R_p])^*]$ and $\sigma(C[]) \setminus \sigma(R_p) \neq \emptyset$ (the context $C'[]$ is just the rest of R after $C[]$ has been defined). If no such context exists, either there is no star containing the subexpression R_p, in which case this path can only produce a bounded number of a_{p+1}^d's, or no symbols with indices greater than p are introduced at all. In any case, this path's contribution to M is finite.

If such a context $C[]$ *does* exist, then, by the choice of p, a symbol with index greater than or equal to $p+2$ is introduced, i.e., belongs to $\sigma(C[]) \setminus \sigma(R_p)$. But then the strings will not have the correct form unless symbols with index $p+1$ are introduced at the same time. However, since $C[]$ is chosen to be the *smallest* context fulfilling the requirements listed above, it is not possible to produce an unbounded number of symbols with index $p+1$ *in between* symbols with index $p+2$, so again only finitely many strings from M can originate from this path. □

Theorem 2 Let R be a regular expression with back referencing, and assume that $L(R) = L(x_k)$. Then R has nested level k.

Proof Let R be written as described in lemma 1. We prove that if some R_i, $i \in \{1, \ldots, k\}$, is *not* of the form $R_i = C_i^A[(C_i^B[{R_{i-1}}^*])\%\alpha]$, for some α, then we get a contradiction. Clearly, this implies that all of these k R_i's are of this form, so the nested level is k.

Now, assume to the contrary that R_i is not of the form described above.

We make some transformations on the expression ${R_i}^*$ (in the remaining part of this proof, we ignore the rest of R). These transformations have the property that if an expression F is transformed into F', then $L(F') \subseteq L(F)$. First, we perform the following transformations on R_{i-1}:

- Replace an alternation with one of its arguments; if a_{i-1}^d, for some $d \in \{l, m, r\}$, is in the signature of one argument and not the other, choose the former. Otherwise, decide arbitrarily.
- Replace a star with its argument.

The transformations are carried out repeatedly until none of them can be applied any more. After this, let E' refer to the expression that R_{i-1} has been transformed into.

Now the remaining parts of ${R_i}^*$ are transformed:

- Replace an alternation with one of its arguments; if E' is a subexpression of one of the arguments, that argument is chosen. Otherwise, decide arbitrarily.
- Replace a star with its argument whenever E' is not a subexpression of that argument.

The transformations are carried out repeatedly until none of them can be applied any more.

The resulting expression has no alternations, and only stars having E' as a subexpression. Since, by assumption, there were no namings between R_{i-1} and R_i, this means that in the transformed expression, subexpressions of namings only contain concatenation as an operator. Now replace all names with the subexpression which has been given that name, and remove the naming. Since the named subexpressions only contain concatenation, each subexpression is in fact simply a string, so this transformation preserves the semantic meaning of the expression. After this, let F refer to the transformed expression. Thus, F is of the form $(C'[{E'}^*])^*$.

Since $L(F) \subseteq L({R_i}^*)$ and since ${R_i}^*$ is a subexpression of R, the strings in $L(F)$ must be of the form

$$x \cdot a_i^l y_1 a_i^m y_1 a_i^r \cdot a_i^l y_2 a_i^m y_2 a_i^r \cdot a_i^l y_3 a_i^m y_3 a_i^r \cdot \cdots \cdot z$$

where $a_i^l \notin \sigma(x)$, $a_i^r \notin \sigma(z)$ and x, z, and the y_i's are some strings.

Since stars are only applied to expressions containing E' as a subexpression, there must exist a constant p such that $L(F)$ intersected with

$$(\Sigma \setminus \{a_i^l\})^* \cdot a_i^l T^* a_i^m T^* a_i^r \cdot a_i^l T^* a_i^m T^* a_i^r \cdot \cdots \text{ (p times) } \cdot a_i^l T^* a_i^m T^* a_i^r \cdot (\Sigma \setminus \{a_i^r\})^*,$$

where $T = \{a_0^l, a_0^m, a_0^r, \ldots, a_{i-1}^l, a_{i-1}^m, a_{i-1}^r\}$, is still infinite. This is seen as follows: Let E_s be the smallest expression containing E' as a subexpression such that R_i can be written $R_i = C_s[E_s]$ and $\{a_i^l, a_i^m, a_i^r\} \subseteq \sigma(E_s)$. Since E_s is the *smallest* expression with this property, there cannot be any stars in E_s applied to a subexpression containing any of the a_i^d's. If there were, strings would not have the correct form (there would be occurrences of one of the a_i^d's without both of the other a_i^d's appearing in between). Removing all stars in $C_s[]$ (in other words changing all stars to a 1 for "one iteration"), would give us a regular expression with a bounded number, p, of a_i^d's. Since E_s contains a star and $\sigma(E') \neq \emptyset$, the language is still infinite.

We now apply a sequential transducer [4] to the result of this intersection. The transducer outputs ε until it meets the first a_i^l after which it outputs whatever it reads. In this way, we can get rid of the x's and still have a regular language. Since the reverse of a regular language is also regular, we can with a similar transducer remove the z's. Again using a sequential transducer, we change every symbol with a subscript different from i to a new symbol b. The strings in this regular language are of the form

$$a_i^l y_1 a_i^m y_1 a_i^r \cdot a_i^l y_2 a_i^m y_2 a_i^r \cdot \cdots \cdot a_i^l y_p a_i^m y_p a_i^r \cdot$$

where the y_i's only contain b's. Applying the pumping lemma [5] to this infinite language shows that it is *not* regular, and we have obtained a contradiction. □

Corollary 3 If $\mathcal{L}_k$ is the set of languages which can be expressed by a regular expression with back referencing of at most k nested levels, then

$$\forall i, j \in I\!N\colon\ i < j \Rightarrow \mathcal{L}_i \subseteq \mathcal{L}_j \ \wedge\ \mathcal{L}_i \neq \mathcal{L}_j. \qquad \square$$

Acknowledgment

The author would like to thank Peter Høyer for many interesting discussions on this and related issues.

References

[1] A. V. Aho, Algorithms for Finding Patterns in Strings, in *Handbook of Theoretical Computer Science, Vol. A*, J. van Leeuwen, ed., Chap. 5, 1073 – 1156: Algorithms and Complexity (Elsevier Science Publishers, Amsterdam, 1990).

[2] D. Angluin, Finding Patterns Common to a Set of String, *Proceedings of the 11th Annual ACM Symposium on Theory of Computing* (1979) 130 - 141.

[3] H. P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics* (North-Holland, Amsterdam, 1981).

[4] M. A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Reading, Massachusetts, 1978).

[5] J. C. Martin, *Introduction to Languages and the Theory of Computation* (McGraw-Hill, New York, 1991).