Tik-79.148 **Spring2001**
Introduction to Theoretical Computer Science
Tutorial 3
Solutions to Demonstration Exercises

The regular expressions are defined inductively:

- $\emptyset$ and all $a \in \Sigma$ are regular expressions.
- If α and β are regular expressions, then also $(\alpha\beta), (\alpha \cup \beta)$ and α^* are regular expressions.
- There are no other regular expressions.

For the regular expression $\emptyset^*$ the shorthand e is used. Usually the extra parenthesis are left out: $((a(bc)) \cup (cd^*)) = abc \cup cd^*$.

A regular expression α defines a language $L(\alpha)$, which is a subset of Σ^*. For instance:

$$L(ab \cup ac^*d) = \{ab, ad, acd, accd, accd, acccd, accccd, \ldots\} \ .$$

If there is no danger for confusion[1], we don't use $L()$ around the regular expression, and the expression itself is used to mean the set of strings (words), which belong to the language it describes.

The regular expressions are perhaps best thought of as patterns. A string is part of the language if it fits the pattern given by the expression:

- $ab \Rightarrow$ we first taken an a, then a b
- $a \cup b \Rightarrow$ we either take an a or a b
- $a^* \Rightarrow$ we take as many a:s as we want (also 0).

The book does properly make clear the distinction between $\emptyset$ and $\emptyset^*$. It goes on to use e to signify the latter, without defining it more accurately. According to the definition $L(\emptyset) = \emptyset$, i.e. the empty language, which does not contain any word. Correspondingly $L(\emptyset^*) = L(\emptyset)^*$, i.e. a language which is constructed in the following way:

$$L(\emptyset)^* = \{w \mid w = w_1 \circ w_2 \circ \cdots \circ w_3, \text{ for some } k \geq 0 \text{ ja } w_1, \ldots, w_k \in \emptyset\}$$

Because no string belongs to the empty set, the only possible value for k is zero. Because by the definition, the concatenation of zero characters is the empty string e, the language becomes

$$L(\emptyset^*) = \{e\}$$

[1] And, unfortunately, many times when there is danger for confusion, as in exercise 4. part b and c.

In other words, the regular expression $\emptyset$ refers to a language which contains no strings, and the expression $\emptyset^*$ refers to a language, which only contains the empty string e.

The union of regular expressions can be denoted in many ways. The most common are $\cup$, $+$ and $|$. In most practical implementations additional operators are defined, to simplify the expressions. The expressiveness of the expressions (the different languages they can describe) usually stays the same.

Regular expressions are used for the syntactic analysis in compilers for programming languages and finding text of certain form from a text file (grep) among other things.

4 a) The statement $baa \in a^*b^*a^*b^*$ is true, because by choosing one of the first b-letters and two of the second a-letters we get the string baa.

b) The statement $b^*a^* \cap a^*b^* = a^* \cup b^*$ is true, because

$$b^*a^* \cap a^*b^* =$$
$$\{e, b^+, a^+, b^+a^+\} \cap \{e, a^+, b^+, a^+b^+\} =$$
$$\{e, a^+, b^+\} = a^* \cup b^*$$

Here $a^+ = aa^*$

Note that the expression on the left hand side of the equation is **not** a regular expression, because intersection cannot appear in a regular expression. Here we refer to the intersection of two regular languages.

c) The statement $a^*b^* \cap c^*d^* = \emptyset$ is false, because

$$a^*b^* \cup c^*d^* =$$
$$\{e, b^+, a^+, b^+a^+\} \cap \{e, c^+, d^+, c^+d^+\} =$$
$$\{e\} \neq \emptyset$$

d) The statement $abcd \in (a(cd)^*b)^*$ is also false, because all strings accepted by the regular expression (except for the empty string) end in the letter b.

5 We show that $a(b \cup c) = ab \cup ac$. By definition the expression $b \cup c$ describes the set $\{b, c\}$. From the definition of concatenation it follows that $a(b \cup c)$ is the set $\{a\} \circ \{b, c\} = \{ab, ac\}$.

6 Claim: If a language L is regular, then also the language

$$L' = \{w \mid uw \in L \text{ for some string } u\}$$

is regular.

L' contains all the strings which appear as the suffixes of the string in L. For instance: if

$$L = \{a, aba, abb\}, \text{ then}$$
$$L' = \{e, a, b, ba, bb, aba, abb\}$$

Here we construct a systematic way to construct a regular expression R', which defines the language L', given the regular expression R, which defines L.

Formally, we denote by S the set of all regular expressions. Now, we wish to find a function $f : S \to S$, such that $f(R) = R'$.

We define f recursively:

i) (base case) $f(\emptyset) = \emptyset$ and $f(a) = a \cup e, a \in \Sigma \cup \{e\}$. In other words, if a regular expression R is empty it has not suffix, and if it is only a letter from the alphabet or e, R' is the union of e and the respective letter.

ii) $f(\alpha \cup \beta) = f(\alpha) \cup f(\beta), \alpha, \beta \in S$. In other words, if R is the union of two regular expressions α and β, the suffixes of the strings the expression defines can have anything either of the subexpressions can have.

iii) $f(\alpha^*) = f(\alpha)\alpha^*$, i.e at the end of the word, the subexpression under the influence of the Kleene star can appear wholly any number of times, and before it something in which the subexpression can end in.

iv) $f(\alpha\beta) = f(\alpha)\beta \cup f(\beta)$, i.e. at the end of word formed by concatenation, there can be either something from the end of the latter part, or the latter part appear wholly and before it is a string in which the first part can end.

After this, only a formal proof of correctness of the rules above is needed. This will not be given here but it can done by showing the correctness of each rule using induction.

Example: Let $R = ba(aa \cup bab)^*$. Now

$$
\begin{aligned}
f(R) &= f(ba(aa \cup bab)^*) \\
&= f(ba)(aa \cup bab)^* \cup f((aa \cup bab)^*) \\
&= (f(b)a \cup f(a))(aa \cup bab)^* \cup f(aa \cup bab)(aa \cup bab)^* \\
&= ((e \cup b)a \cup (e \cup a))(aa \cup bab)^* \cup (f(aa) \cup f(bab))(aa \cup bab)^* \\
&= (e \cup a \cup ba)(aa \cup bab)^* \cup ((f(a)a \cup f(a)) \cup (f(b)ab \cup f(a)b \cup f(b)))(aa \cup bab)^* \\
&= (e \cup a \cup ba)(aa \cup bab)^* \cup (e \cup a \cup aa \cup b \cup ab \cup bab)(aa \cup bab)^* \\
&= (e \cup a \cup aa \cup b \cup ab \cup bab \cup ba)(aa \cup bab)^* \\
&= (e \cup a \cup b \cup ab \cup ba)(aa \cup bab)^*
\end{aligned}
$$