PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

Complexity Theory, Semester I 2017 - IIC3242
Homework 2
Deadline: Tuesday, April 4th, 2017

1 A cool reduction [7 points]

Usual regular expressions use the operators of union, concatenation and Kleene star to define sets of words over some finite alphabet Σ. In this problem we will explore what happens when we extend these expressions with two additional operators: intersection and mixing.

An *extended regular expression (ER)* over an alphabet Σ is defined as follows.

1. ε is an ER;
2. Every $a \in \Sigma$ is an ER;
3. If e_1 and e_2 are ERs, then so is $e_1 + e_2$ (union);
4. If e_1 and e_2 are ERs, then so is $e_1 \cdot e_2$ (concatenation);
5. If e_1 and e_2 are ERs, then so is $e_1 \cap e_2$ (intersection);
6. If e_1 and e_2 are ERs, then so is $e_1 \& e_2$ (mixing); and
7. If e is an ERs, then so is e^* (Kleene star).

Every extended regular expression e defines a set of words $L(e)$ in the following way:

1. If $e = \varepsilon$ then $L(e) = \{\varepsilon\}$;
2. If $e = a \in \Sigma$ then $L(e) = \{a\}$;
3. If $e = e_1 + e_2$ then $L(e) = L(e_1) \cup L(e_2)$;
4. If $e = e_1 \cdot e_2$ then $L(e) = \{w | w = w_1 \cdot w_2 \text{ and } w_1 \in L(e_1), w_2 \in L(e_2)\}$;
5. If $e = e_1 \cap e_2$ then $L(e) = L(e_1) \cap L(e_2)$;
6. If $e = e_1 \& e_2$ then $L(e) = \{w_1 \& w_2 | w_1 \in L(e_1), w_2 \in L(e_2)\}$; where the mixing of two words x, y over Σ, denoted $x \& y$, is defined as the *set of all words* of the form $x_1 \cdot y_1 \cdots x_k \cdot y_k$, where:
   - $k > 0$ and
   - x_i, y_i are words over Σ (they can be ε) and
   - $x = x_1 \cdot x_2 \cdots x_k$ and
   - $y = y_1 \cdot y_2 \cdots y_k$.
7. If $e = e_1^*$ then $L(e) = \{w_1 \cdot w_2 \cdots w_k | k \geq 1 \text{ and } w_i \in L(e_1) \text{ for } i = 1 \ldots k\} \cup \{\varepsilon\}$.

To give an example of how the new operations work consider the alphabet $\Sigma = \{a, b, c\}$ and an expression $e = ab\&cca$. Then we have that $accba \in L(e)$, since we can decompose $x = ab$ as $x_1 = a$ and $x_2 = b$; and we can decompose $y = cca$ as $y_1 = cc$ and $y_2 = a$. Similarly we have that $accab \in L(e)$, but this time $y = cca$ is decomposed as $y_1 = cca$ and $y_2 = \varepsilon$. It is also easy to check that e.g. $cbaca$ does not belong to $L(e)$, since it does not have an a before a b, thus it is not possible to construct the word ab. Similarly $abc\&(\Sigma \cup \varepsilon)^n$ will mix any length n word over Σ into abc.

We define the following problem:

$$\text{MEMBERSHIP} = \{\langle \Sigma, e, w \rangle |\ e \text{ is an ER over } \Sigma \text{ and } w \in L(e)\}.$$

By giving a reduction from the problem 3SAT show that MEMBERSHIP(Σ) is NP-hard (6.5 points). Notice that one input to MEMBERSHIP is the alphabet Σ. Argue why this is not necessary and why you can prove NP-hardness even for one particular finite alphabet (0.5 points).

Hint: It's easy. You might want to make heavy use of ε and intersection. It is possible to use the mixing operator | only once (although you are allowed to use it as many times as you wish). One way is to try and check the membership of a word $v_1 \cdot v_2 \cdots v_k$, which is simply a concatenation of all the variables appearing in a 3CNF formula (assuming Σ equals the set of variables in your formula). You could then, for each clause i, define an expression C_i which contains all words w of length at most n such that: (1) at least one positive literal from C_i appears as a symbol in w; or (2) there is at least one negative literal $\neg v_i$ in C_i such that v_i does *not* appear in w. From here it is quite easy to get the required expression using intersections and interleaving.