1 Regularity and Language Operations (20)

Background
We have seen that regular languages are closed under the operations of union, concatenation and Kleene star. Going in the opposite direction is a bit more complicated.

Task
Determine whether the following propositions are true or false. Provide a clear proof for your claim in each case.

A. $L_1 \cup L_2$ regular and $|L_1| = 1$ implies $L_2$ regular.
B. $L_1 \cup L_2$ regular and $L_1$ finite implies $L_2$ regular.
C. $L_1 \cup L_2$ regular and $L_1$ regular implies $L_2$ regular.
D. $L_1 \cdot L_2$ regular and $|L_1| = 1$ implies $L_2$ regular.
E. $L_1 \cdot L_2$ regular and $L_1$ finite implies $L_2$ regular.
F. $L_1 \cdot L_2$ regular and $L_1$ regular implies $L_2$ regular.
G. $L^*$ regular implies $L$ regular.

Comment
If you construct a counterexample, make it small.

2 Word Binomials (40)

Background
By a subsequence of a word $v = v_1v_2 \ldots v_m$ we mean any word $u = v_{i_1}v_{i_2} \ldots v_{i_r}$ where $1 \leq i_1 < i_2 < \ldots i_r \leq m$ is a strictly increasing sequence of indices. Thus $bbc$ and $cab$ are subsequences of $ababacaba$ but $cbb$ is not.
Note that a specific word can occur multiple times as a subsequence of another. For example, \( aab \) appears 7 times in \( ababacaba \). We write

\[
\binom{v}{u} = C(v, u) = \text{number of occurrences of } u \text{ as a subsequence of } v.
\]

The notation is justified since “word binomials” generalize ordinary binomial coefficients: \( \binom{n}{k} = \binom{\alpha}{\alpha^k} \).

**Task**

Let \( \delta_{a,b} = 1 \) iff \( a = b \), 0 otherwise, and \( a, b \in \Sigma \) and \( u, v, u_i, v_i \in \Sigma^* \).

A. Show that

\[
\binom{va}{ub} = \binom{v}{ub} + \delta_{a,b} \binom{v}{u}
\]

B. Show that

\[
\binom{v_1v_2}{u} = \sum_{u = u_1u_2} \binom{v_1}{u_1} \binom{v_2}{u_2}
\]

C. Give an efficient algorithm to compute word binomials.

D. Give a simple description (in terms of union, concatenation and Kleene star) of the language

\[
L = \{ v \in \{a,b\}^* | \binom{v}{ab} = 3 \}
\]

E. Construct the minimal DFA for \( L \).

F. Generalize: given a word \( u \) and an integer \( r \) construct a DFA that accepts

\[
L(u, r) = \{ v \in \Sigma^* | \binom{v}{u} = r \}
\]

Is your machine always minimal?

---

### 3 Window Languages (40)

**Background**

In this problem we only consider languages in \( \Sigma^+ \), so the empty word causes no technical problems. A language \( L \) is a window language if membership in \( L \) can be tested by sliding a window of size 2 across the word and observing the 2-factors of the word.

Here is a formalization of this idea. Define \( \text{fact}_2(x) = \{ ab \in \Sigma^2 | x = uabv, u, v \in \Sigma^* \} \) to be the set of all 2-factors of \( x \). Define an equivalence relation \( \approx \) on \( \Sigma^+ \) as follows:

\[
x \approx y \iff x_1 = y_1 \land \text{fact}_2(x) = \text{fact}_2(y) \land x_{-1} = y_{-1}
\]

Then \( L \) is a window language if it saturates \( \approx \): \( L = \bigcup_{x \in L} [x] \). Write \( \Sigma^{++} \) for all words of length at least 2. Given \( F \subseteq \Sigma^2 \) let \( L_F = \{ x \in \Sigma^{++} | \text{fact}_2(x) \subseteq F \} \).
Task

A. Find a fast algorithm to check whether $L_F$ is finite. What is the running time of your algorithm?

B. Find a simple, infinite language $L \subseteq \Sigma^{++}$ that is not of the form $L_F$.

C. Show that $\approx$ is a congruence: an equivalence relation such that $x \approx y$ implies $uxv \approx uyv$ for all $u, v \in \Sigma^*$.

D. Show that every window language is regular.

E. Show that every regular language is a homomorphic image of a window language.

Comment

For the last part you need to produce a regular language $R \subseteq \Gamma^*$ for some alphabet $\Gamma$ and a homomorphism $\Phi : \Gamma^* \rightarrow \Sigma^*$ such that $\Phi(R) = L$. $\Gamma$ will depend strongly on a DFA for the given regular language.

This is perhaps a little counterintuitive: the window seems to be too narrow for arbitrary regular languages with long-distance constraints (say, every $a$ is followed by a $b$ after at most 123 letters).