apl. Prof. Dr. Thomas Noll Philipp Berger, Matthias Volk

Compiler Construction 2018/19
— Exercise Sheet 2 —

Hand in until October 29th before the exercise class.

General Remarks

- If you have questions regarding the exercises and/or lecture, feel free to post in the L2P forum, write us an email at cb2018@i2.informatik.rwth-aachen.de or visit us at our office.

Exercise 1 (8 Points)

Let $\Sigma := (\texttt{a}|\ldots|\texttt{z})$ be the set of letters, $N := (\texttt{0}|\texttt{1}|\ldots|\texttt{9})$ the set of digits, and $\Omega := \Sigma \cup N \cup \{\texttt{+}, \texttt{-}, \texttt{.}\}$ the set of all symbols.

Consider the four regular expressions

$$\begin{aligned}\alpha_1 &= \texttt{-}\\ \alpha_2 &= \texttt{+}\\ \alpha_3 &= \Sigma^+\\ \alpha_4 &= N \mid N\texttt{.}N \mid N\texttt{.}N\texttt{e}(\texttt{-} \mid \texttt{+})N^+\end{aligned}$$

Hint: The symbols `.` in α_4 corresponds to the dot character and is *not* the concatenation symbol.

(a) For each regular expression $\alpha_i, i = 1, \ldots, 4$ construct a DFA $\mathfrak{A}_i$ such that $\mathcal{L}(\mathfrak{A}_i) = [\![\alpha_i]\!]$.

(b) Construct the product automaton $\mathfrak{A} = \mathfrak{A}_1 \otimes \mathfrak{A}_2 \otimes \mathfrak{A}_3 \otimes \mathfrak{A}_4$.
Hint: It suffices to depict the reachable fragment of the product automaton.

(c) Determine the *first match* partitioning of the set of final states in $\mathfrak{A}$ for the ordering $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$.

(d) Determine the set of reachable and productive states in $\mathfrak{A}$.

(e) Compute the run of the corresponding backtracking DFA for input `-0.2e+fa`. Provide the run by giving the corresponding configurations.

Exercise 2 (12 Points)

On July 20, 2016 the popular website stackoverflow.com experienced an unexpected outage due to a malformed post that caused its backtracking Regex engine to consume extraordinary high CPU time.[1] This is an example of a *regular expression denial-of-service* (ReDoS) attack: An attacker tries to paralyze an application by feeding it with input strings that exhibit high computational complexity.

In this exercise, we have a closer look at such "attack strings". To this end, we execute the FLM analysis based on the NFA method from the lecture (lecture 4, slide 10).

Furthermore, we call an NFA $\mathfrak{A}$ *vulnerable* if and only if the worst-case complexity of the above FLM analysis is exponential in the length of the input string.

Hint: In this exercise, we assume that our FLM analysis algorithm is very naive and explores each possible run of an NFA on a word individually.

(a) Provide an example of a vulnerable NFA $\mathfrak{A}$ and, for each $k \geq 1$, an input string w_n of length $|w_n| = n \geq k$ such that the runtime of the FLM analysis on $\mathfrak{A}$ and w_n is exponential in n.

[1] Details are provided at `http://stackstatus.net/post/147710624694/outage-postmortem-july-20-2016`

(b) A *path* in an NFA $\mathfrak{A} = (Q, \Omega, \delta, q_0, F)$ is a sequence of transitions $\pi = (q_1, \ell_1, q_2) \dots (q_{m-1}, \ell_{m-1}, q_m)$ such that $q_i \in Q$, $\ell_i \in \Omega$, and $q_{i+1} \in \delta(q_i, \ell_i)$. Moreover, we set $\mathit{labels}(\pi) = \ell_1 \dots \ell_{m-1}$.

Show that an NFA $\mathfrak{A}$ is vulnerable if there exists a state q and two distinct paths π_1, π_2 such that

(i) both π_1 and π_2 start and end at state q,

(ii) $\mathit{labels}(\pi_1) = \mathit{labels}(\pi_2)$,

(iii) there is a path π_p from initial state q_0 to q, and

(iv) there is a path π_s from q to a state $q_r \notin F$.

(c) Sketch a procedure that takes an NFA $\mathfrak{A}$ and one of its states q as an input and checks whether $\mathfrak{A}$ is vulnerable for the given state q.