ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theoInf20

Exemplary Solutions – Sheet 13

Zürich, December 11, 2020

Solution to Exercise 35

(a) Assume that L_1 is context-free. Then we consider the integer $n_L = n_{L_1}$ from the pumping lemma for context-free languages and the word $z = a^{n_L}b^{{n_L}^3} \in L_1$. The pumping lemma yields a decomposition $z = uvwxy$ that satisfies the three conditions of the pumping lemma. In particular, $|vx| \geq 1$ and $|vwx| \leq n_L$.

We perform a case distiction with respect to the position of v and x in z.

If $vx = a^i$ for some $i \in \mathbb{N} - \{0\}$, then we consider the word $z_2 = uv^2wx^2y$. It follows that $z_2 = a^{n_L+i}b^{{n_L}^3} \notin L_1$ as $(n_L + i)^3 \neq n_L^3$. This yields a contradiction to the condition (iii) of the pumping lemma stating that $\{uv^iwx^iy \mid i \in \mathbb{N}\} \subseteq L_1$.

If $vx = b^i$ for some $i \in \mathbb{N} - \{0\}$, a contradiction can be derived analogously.

If $vx = a^ib^j$ for $i, j \in \mathbb{N} - \{0\}$, then $j \leq n_L$ because of $|vwx| \leq n_L$. We again consider the word $z_2 = uv^2wx^2y$. This word contains $(n_L + i)$ many a's and $n_L^3 + j$ many b's. The fact that $(n_L + i)^3 = n_L^3 + 3n_L^2i + 3n_Li^2 + i^3 > n_L^3 + n_L \geq n_L^3 + j$ yields a contradiction to the condition (iii) of the pumping lemma stating that $\{uv^iwx^iy \mid i \in \mathbb{N}\} \subseteq L_1$.

We derived a contradiction in each of the possible cases, thus our assumption is false and L_1 is not context-free.

(b) Assume that L_2 is context-free. Then we consider the integer $n_L = n_{L_2}$ from the pumping lemma for context-free languages and the word $z = a^{n_L}b^{n_L}ca^{n_L}b^{n_L} \in L_2$. The pumping lemma yields a decomposition $z = uvwxy$ that satisfies the three conditions of the pumping lemma. In particular, $|vx| \geq 1$ and $|vwx| \leq n_L$.

We perform a case distiction with respect to the position of v and x in z.

If vx contains the symbol c, then $z_0 = uwy$ contains no c. Hence, $z_0 \notin L_2$ which is a contradiction to the condition (iii) of the pumping lemma.

If vx only contains symbols from the prefix $a^{n_L}b^{n_L}$ of z, then the subword of $z_2 = uv^2wx^2y$ before the occurrence of c is longer than the subword behind it. Hence, $z_2 \notin L_2$ which is a contradiction to the condition (iii) of the pumping lemma.

If vx only contains symbols from the suffix $a^{n_L}b^{n_L}$ of z, then the subword of $z_0 = uwy$ before the occurrence of c is longer than the subword behind it. Hence, $z_0 \notin L_2$ which is a contradiction to the condition (iii) of the pumping lemma.

It remains to consider the case that vx contains symbols before as well as after the occurrence of c. Since $|vwx| \leq n_L$, vx cannot contain any a's from the prefix a^{n_L} and cannot contain any b's from the suffix b^{n_L}. Hence, $z_2 = uv^2wx^2y$ contains more b's before the occurrence of c than behind it. Hence, $z_2 \notin L_2$ which is a contradiction to the condition (iii) of the pumping lemma.

We derived a contradiction in each of the possible cases, thus our assumption is false and L_2 is not context-free.

Solution to Exercise 36

As L_1 is context-free and L_2 is regular, L_1 is accepted by a pushdown automaton $M_1 = (Q_1, \{a, b\}, \Gamma, \delta_1, q_{0,1}, Z_0)$ and L_2 is accepted by a (nondeterministic) finite automaton $M_2 = (Q_2, \{a, b\}, \delta_2, q_{0,2}, F)$.

Now we can essentially run both automata in parallel, using the product construction just like when combining two finite automata. The only additional difficulty is that M_1 accepts once the stack becomes empty and M_2 accepts upon reaching an accepting state. The new automaton being a pushdown automaton has to accept by emptying the stack.

To this end, we only allow to empty the stack if M_2 simultaneously makes a transition to an accepting state. To be able to recognize that the stack becomes empty, we have to guarantee that Z_0 is only used once at the very bottom of the stack. To this end, we first transform M_1 into an equivalent pushdown automaton $M_1' = (Q_1', \{a, b\}, \Gamma', \delta_1', q_{0,1}', Z_0')$ where

$$\begin{aligned}
Q_1' &= Q_1 \cup \{q_{0,1}'\}, q_{0,1}' \notin Q_1, \\
\Gamma' &= \Gamma \cup \{Z_0'\}, Z_0' \notin \Gamma, \\
\delta_1'(q_{0,1}', \lambda, Z_0') &= \{(q_{0,1}, Z_0'Z_0)\}, \\
\delta_1'(q, \lambda, Z_0') &= \{(q, \lambda)\} \text{ for all } q \in Q_1, \\
\delta_1'(q, x, Z) &= \delta_1(q, x, Z) \text{ for all } q \in Q_1, x \in \{a, b\} \cup \{\lambda\}, \text{ and } Z \in \Gamma.
\end{aligned}$$

The pushdown automaton M_1' uses Z_0' as the symbol at the very bottom of the stack. In the only possible initial step, it puts Z_0 on top of it and changes its state to $q_{0,1}$, i.e., it reproduces the initial configuration of M_1 (with an additional symbol Z_0' at the very bottom of the stack). Then it behaves exactly like M_1 until Z_0' is read again. In the case when Z_0' is read (i.e., it is at the top of the stack), it is removed from the stack yielding an empty stack and finishing the computation. This happens if and only if everything else has been removed from the stack, i.e., the stack of M_1 has been emptied. Hence, the two automata are equivalent.

Using the product construction on M_1' and M_2, we construct the following pushdown automaton $M = (Q, \{a, b\}, \Gamma', \delta, q_0, Z_0')$ where

$$\begin{aligned}
Q &= Q_1' \times Q_2, \\
q_0 &= (q_{0,1}', q_{0,2}), \\
\delta(q_0, \lambda, Z_0') &= \{((q_{0,1}, q_{0,2}), Z_0'Z_0)\}, \\
\delta((p, q), \lambda, Z_0') &= \{((p, q), \lambda, \lambda)\} \text{ for all } p \in Q_1, q \in F,
\end{aligned}$$

and, for all $(p, q) \in Q$, $x \in \{a, b\}$, and $Z \in \Gamma$, we set

$$\delta((p, q), x, Z) = \{((p', q'), \beta) \mid (p', \beta) \in \delta_1'(p, x, Z), q' \in \delta_2(q, x)\} \text{ as well as}$$

$$\delta((p,q),\lambda,Z) = \{((p',q),\beta) \mid (p',\beta) \in \delta_1'(p,\lambda,Z)\}.$$

The construction implies, for every word $w \in \{a,b\}^*$, that an accepting computation

$$((q_{0,1}',q_{0,2}),w,Z_0') \vdash_M \ldots \vdash_M ((p,q),u,\alpha) \vdash_M \ldots \vdash_M ((p',q'),\lambda,\lambda)$$

of M on w corresponds to the accepting computations

$$(q_{0,1}',w,Z_0') \vdash_{M_1'} \ldots \vdash_{M_1'} (p,u,\alpha) \vdash_{M_1'} \ldots \vdash_{M_1'} (p',\lambda,\lambda)$$

and

$$(q_{0,2},w) \vdash_{M_2} \ldots \vdash_{M_2} (q,u) \vdash_{M_2} \ldots \vdash_{M_2} (q',\lambda)$$

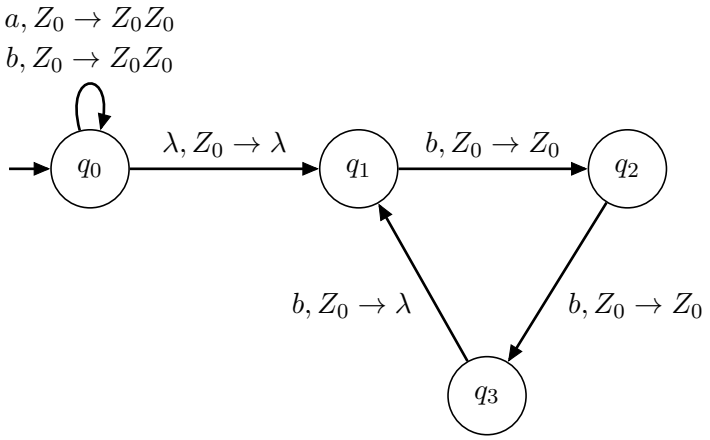
of M_1' and M_2 on w, respectively, and vice-versa. One only has to make sure that the steps of M_1' and M reading an empty word are such that the state of M_2 is left unchanged.
Overall, we obtain

$$L(M) = L(M_1') \cap L(M_2) = L_1 \cap L_2,$$

thus $L_1 \cap L_2$ is context-free.

Solution to Exercise 37

The language L is accepted by the following pushdown automaton M:



The idea behind the construction is as follows: As long as the pushdown automaton M stays in the state q_0, it reads the part u of the input word and writes one symbol Z_0 onto the stack for every input symbol it reads. Taking a λ-transition into the state q_1, M decides nondeterministically that the part v of the input word starts. During one iteration of the loop q_1, q_2, q_3, q_1, M reads three symbols b and removes one symbol from the stack (during the transition from q_3 to q_1).

The correctness of the pushdown automaton can be derived from the following observation: after reading k input symbols in the state q_0, $k+1$ symbols Z_0 are pushed onto the stack. One of them gets removed during the transition from q_0 to q_1 so that exactly k symbols are on the stack once the state q_1 is reached. To remove one of these symbols, M must clearly read three symbols b from the input.

As M accepts the input if and only if the stack is empty once the entire input has been read, every accepting computation must have read $3k$ symbols b in the states q_1, q_2, q_3. Hence, exactly the inputs of the form uv with $|u| = k$ and $v = b^{3k}$ for some $k \in \mathbb{N}$ are accepted.