

Introduction to the Theory of Computation

Jean Gallier

Solutions to the Final Exam

April 26

Problem 1. Since $h: \Sigma^* \rightarrow \Delta^*$ is a homomorphism, using the fact that $h(uv) = h(u)h(v)$ for all $u, v \in \Sigma^*$, it is easy to prove that $h(L_1L_2) = h(L_1)h(L_2)$ and $h(L^*) = (h(L))^*$, for all $L_1, L_2, L \subseteq \Sigma^*$. Moreover, for any function, h , we have $h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$.

Given any regular expression, R , over Σ , we prove that $\mathcal{L}[h(R)] = h(\mathcal{L}[R])$ by induction on the structure of R . In the base case where $R = a \in \Sigma$, as $h(a)$ is a string in Δ^* , we have

$$\mathcal{L}[h(a)] = h(a) = h(\mathcal{L}(a)),$$

as required. The other two base cases are also trivial:

$$\mathcal{L}[h(\emptyset)] = \mathcal{L}[\emptyset] = \emptyset = h(\emptyset) = h(\mathcal{L}[\emptyset])$$

and

$$\mathcal{L}[h(\epsilon)] = \mathcal{L}[\epsilon] = \epsilon = h(\epsilon) = h(\mathcal{L}[\epsilon]).$$

For the induction step, there are three cases:

(1) $R = (S_1 + S_2)$. Then,

$$\begin{aligned} \mathcal{L}[h(S_1 + S_2)] &= \mathcal{L}[(h(S_1) + h(S_2))] \\ &= \mathcal{L}[h(S_1)] \cup \mathcal{L}[h(S_2)] \\ &= h(\mathcal{L}[S_1]) \cup h(\mathcal{L}[S_2]) \\ &= h(\mathcal{L}[S_1] \cup \mathcal{L}[S_2]) \\ &= h(\mathcal{L}[(S_1 + S_2)]). \end{aligned}$$

(2) $R = (S_1 \cdot S_2)$. Then,

$$\begin{aligned} \mathcal{L}[h(S_1 \cdot S_2)] &= \mathcal{L}[(h(S_1) \cdot h(S_2))] \\ &= \mathcal{L}[h(S_1)] \cdot \mathcal{L}[h(S_2)] \\ &= h(\mathcal{L}[S_1]) \cdot h(\mathcal{L}[S_2]) \\ &= h(\mathcal{L}[S_1] \cdot \mathcal{L}[S_2]) \\ &= h(\mathcal{L}[(S_1 \cdot S_2)]). \end{aligned}$$

(3) $R = S^*$. Then,

$$\begin{aligned}
\mathcal{L}[h(S^*)] &= \mathcal{L}[h(S)^*] \\
&= (\mathcal{L}[h(S)])^* \\
&= (h(\mathcal{L}[S]))^* \\
&= h(\mathcal{L}[S]^*) \\
&= h(\mathcal{L}[S^*]).
\end{aligned}$$

Problem 2. By the version of Myhill-Nerode for congruences, there is a left and right invariant equivalence relation of finite index, \sim , with the property that L is the union of some of the equivalence classes of \sim . If we can prove that $L^{1/3}$ is the union of classes of \sim , then by the usual Myhill-Nerode theorem, $L^{1/3}$ will be regular. It is enough to prove that for any $u \in L^{1/3}$, the entire equivalence class, $[u]_{\sim}$, is contained in $L^{1/3}$. So, let $u \in L^{1/3}$ and assume $u \sim v$. By definition, $uuu \in L$. Now, as \sim is a congruence, from $u \sim v$, we deduce $uu \sim vv$, and then

$$uuu \sim vvv.$$

Since L is the union of classes of \sim and $uuu \in L$, we get $vvv \in L$. But, $vvv \in L$ means $v \in L^{1/3}$, as desired. This concludes the proof that $[u]_{\sim} \subseteq L^{1/3}$ and thus, that $L^{1/3}$ is regular.

The problem can also be solved in terms of DFA's; here it is! Let $D = (Q, \Sigma, \delta, q_0, F)$ a DFA accepting L . Observe that if www is accepted by D , then the computation on input www can be broken into three pieces: the first one (on input w) starts from q_0 and ends in some state p ; the second one (also on input w) starts from p and ends in some state q ; the third one (still on input w) starts from q and ends in some final state $f \in F$.

Thus, for all $p, q \in Q$, it is natural to define the language

$$L_{p,q} = \{w \in \Sigma^* \mid \delta^*(q_0, w) = p \wedge \delta^*(p, w) = q \wedge \delta^*(q, w) \in F\}.$$

We claim that

$$L^{1/3} = \bigcup_{p,q \in Q} L_{p,q}.$$

The inclusion $L_{p,q} \subseteq L^{1/3}$ is obvious, and if $www \in L$, as we observed earlier, there are some $p, q \in Q$ such that $w \in L_{p,q}$.

Now, let

$$\begin{aligned}
L_{p,q}^1 &= \{w \in \Sigma^* \mid \delta^*(q_0, w) = p\}, \\
L_{p,q}^2 &= \{w \in \Sigma^* \mid \delta^*(p, w) = q\}, \\
L_{p,q}^3 &= \{w \in \Sigma^* \mid \delta^*(q, w) \in F\}.
\end{aligned}$$

If we define $D_{p,q}^i$, for $i = 1, 2, 3$ as follows

$$\begin{aligned} D_{p,q}^1 &= (Q, \Sigma, \delta, q_0, \{p\}), \\ D_{p,q}^2 &= (Q, \Sigma, \delta, p, \{q\}), \\ D_{p,q}^3 &= (Q, \Sigma, \delta, q, F), \end{aligned}$$

it is obvious that

$$L_{p,q}^i = L(D_{p,q}^i)$$

for $i = 1, 2, 3$. However, it is also obvious that

$$L_{p,q} = L_{p,q}^1 \cap L_{p,q}^2 \cap L_{p,q}^3.$$

Since the $L_{p,q}^i$ are regular and the regular languages are closed under union and intersection,

$$L^{1/3} = \bigcup_{p,q \in Q} L_{p,q}^1 \cap L_{p,q}^2 \cap L_{p,q}^3$$

proves that $L^{1/3}$ is regular.

Problem 3. Consider the following context-free grammar generating

$$L_2 = \{xcy \mid 2|x| = |y|, x, y \in \{a, b\}^*\}.$$

$$\begin{aligned} S &\longrightarrow USUU \\ S &\longrightarrow c \\ U &\longrightarrow a \\ U &\longrightarrow b. \end{aligned}$$

We prove that

$$L_3 = \{xxcx \mid x \in \{a, b\}^*\}$$

is not context-free by contradiction using Ogden's lemma. Let K be the constant of Ogden's lemma and let $w = a^K \mathbf{b} a^K \mathbf{b} c a^K b \in L_3$, with the second group $a^K b$ marked. Then, we can write $w = uvxyz$ and we know that x contains some marked letter.

Case 1. If both u and v contain some marked letter, since x also contains some marked letter, v is completely within the marked block $a^K b$, and so, u contains the first block $a^K b$. Therefore, when we pump v and y up, getting $uv^i xy^i z$, the first block $a^K b$ is unaffected and we get a string not of the form $xxcx$, a contradiction.

Case 2. If both y and z contain some marked letter, since x also contains some marked letter, y is completely within the marked block $a^K b$, and so, z contains the third block $a^K b$. This time, when we pump v and y up, the third block $a^K b$ is unaffected and we get a string not of the form $xxcx$, a contradiction.

Since this exhausts all possibilities for the legal decompositions of w , we conclude that L_3 is not context-free.

Problem 4. Construct a linear context-free grammar, G' , as follows: The nonterminals are triples $(p, A, q) \in Q \times N \times Q$ or S_0 , with S_0 a new symbol and the productions are:

$$S_0 \longrightarrow (q_0, S, f), \quad \text{for every } f \in F;$$

$$(p, A, q) \longrightarrow u \quad \text{iff} \quad A \longrightarrow u \in P \quad \text{and} \quad \delta^*(p, u) = q,$$

for all $p, q \in Q$;

$$(p, A, q) \longrightarrow u(r, B, s)v \quad \text{iff} \quad A \longrightarrow uBv \in P, \quad \delta^*(p, u) = r \quad \text{and} \quad \delta^*(s, v) = q,$$

for all $p, q, r, s \in Q$.

First, we prove by induction on n

Claim 1:

$$(p, A, q) \xRightarrow{n} x(r, B, s)y \quad \text{iff} \quad A \xRightarrow{n} xBy, \quad \delta^*(p, x) = r \quad \text{and} \quad \delta^*(s, y) = q,$$

for all $x, y \in \Sigma^*$, all $p, q, r, s \in Q$ and all $A, B \in N$.

When $n = 0$, we have $x = y = \epsilon$, $p = r$, $s = q$ and $A = B$ and the claim holds trivially. Assume the induction hypothesis holds for n and consider the derivation

$$(p, A, q) \xRightarrow{n} x(r, B, s)y \Longrightarrow xu(r', C, s')vy.$$

By definition of the productions, $B \longrightarrow uCv \in P$, $\delta^*(r, u) = r'$ and $\delta^*(s', v) = s$. By the induction hypothesis,

$$A \xRightarrow{n} xBy, \quad \delta^*(p, x) = r \quad \text{and} \quad \delta^*(s, y) = q.$$

It follows that

$$A \xRightarrow{n} xBy \Longrightarrow xuCvy$$

and also that

$$\delta^*(p, xu) = \delta^*(\delta^*(p, x), u) = \delta^*(r, u) = r'$$

and

$$\delta^*(s', vy) = \delta^*(\delta^*(s', v), y) = \delta^*(s, y) = q,$$

establishing the induction hypothesis. Conversely, assume

$$A \xRightarrow{n} xBy \Longrightarrow xuCvy, \quad \delta^*(p, xu) = r' \quad \text{and} \quad \delta^*(s', vy) = q.$$

Let $r = \delta^*(p, x)$ and $s = \delta^*(s', v)$, so that $\delta^*(r, u) = r'$ and $\delta^*(s, y) = q$. By definition, $(r, B, s) \longrightarrow u(r', C, s')v$ is a production and by the induction hypothesis,

$$(p, A, q) \xRightarrow{n} x(r, B, s)y.$$

Consequently, we get the derivation

$$(p, A, q) \xRightarrow{n} x(r, B, s)y \implies xu(r', C, s')vy,$$

establishing the induction hypothesis and Claim 1.

Now, we prove by induction on n

Claim 2:

$$(p, A, q) \xRightarrow{n} w \quad \text{iff} \quad A \xRightarrow{n} w \quad \text{and} \quad \delta^*(p, w) = q,$$

for all $w \in \Sigma^*$, all $p, q \in Q$ and all $A \in N$.

The base case is $n = 1$. By definition of the productions, this case is trivial.

Assume the induction hypothesis holds for n and consider the derivation

$$(p, A, q) \xRightarrow{n} x(r, B, s)y \implies xuy.$$

By definition of the productions, we have $B \longrightarrow u \in P$ and $\delta^*(r, u) = s$. By Claim 1,

$$A \xRightarrow{n} xBy, \quad \delta^*(p, x) = r \quad \text{and} \quad \delta^*(s, y) = q.$$

It follows that

$$A \xRightarrow{n} xBy \implies xuy$$

and also that

$$\delta^*(p, xuy) = \delta^*(\delta^*(p, x), uy) = \delta^*(r, uy) = \delta^*(\delta^*(r, u), y) = \delta^*(s, y) = q,$$

establishing the induction hypothesis. Conversely, assume

$$A \xRightarrow{n} xBy \implies xuy \quad \text{and} \quad \delta^*(p, xuy) = q.$$

Let $r = \delta^*(p, x)$ and $s = \delta^*(r, u)$, so that $\delta^*(s, y) = q$. By definition, $(r, B, s) \longrightarrow u$ is a production and by Claim 1

$$(p, A, q) \xRightarrow{n} x(r, B, s)y.$$

It follows that

$$(p, A, q) \xRightarrow{n} x(r, B, s)y \implies xuy$$

and Claim 2 is proved.

(ii) If we apply Claim 2 to (q_0, S, f) , where $f \in F$, we get

$$S_0 \implies (q_0, S, f) \xRightarrow{*} w \quad \text{iff} \quad S \xRightarrow{*} w \quad \text{and} \quad \delta^*(q_0, w) = f,$$

for all $w \in \Sigma^*$ and all $f \in F$. Therefore, $w \in L(G')$ iff $w \in L(G) \cap L(D) = L \cap R$, i.e., $L \cap R$ is linear context-free.

Problem 5. Let $A = \{i \in \mathbb{N} \mid \varphi_i(0) = 1\}$. If we prove that A is nontrivial, then by Rice's theorem, A is not recursive, which is what we need to prove. Now, the constant function with

value 1 is partial recursive (in fact, primitive recursive, it is $S \circ E$), so $A \neq 0$. Furthermore, $P_1^1(0) = 0 \neq 1$, so $A \neq \mathbb{N}$. Therefore, A is nontrivial.

Let $B = \{\in \mathbb{N} \mid \varphi_i \text{ is undefined for a finite number of input values}\}$. Again, using Rice's theorem, it is enough to show that B is nontrivial. Since the zero function is defined for all inputs, $B \neq \emptyset$. On the other hand, since there is partial recursive function undefined everywhere, $B \neq \mathbb{N}$. Therefore, B is nontrivial

Problem 6. Every r.e. set, L , is accepted by a Turing machine, M , so that if $w \in L$, then M halts in a proper halting ID with output 1, and if $w \notin L$, then either M halts in a proper halting ID with output 0, or M diverges on input w . We can modify M so that it loops forever if it halts with output 0.

From the class notes (Section 6.9), we know that the language, L_M , consisting all of strings

$$w_0 \# w_1^R \# w_2 \# w_3^R \# \cdots \# w_{2k} \# w_{2k+1}^R,$$

or

$$w_0 \# w_1^R \# w_2 \# w_3^R \# \cdots \# w_{2k-2} \# w_{2k-1}^R \# w_{2k},$$

where $k \geq 0$, w_0 is a starting ID, $w_i \vdash w_{i+1}$ for all i with $0 \leq i < 2k + 1$ and w_{2k+1} is proper halting ID in the first case, $0 \leq i < 2k$ and w_{2k} is proper halting ID in the second case, is of the form $L_1 \cap L_2$, for two context-free languages L_1 and L_2 . Recall that the starting ID, w_0 , is of the form $q_0 w$, where w is the input string and q_0 is the start state. We need to apply a homomorphism that deletes all symbols in the encoding, $w_0 \# w_1^R \# \cdots \# w_{2k+1}^R$ (or $w_0 \# w_1^R \# \cdots \# w_{2k}$) of a computation, except for w in $w_0 = q_0 w$. For this, create a new alphabet consisting of a copy of Γ (disjoint from Γ), say $\tilde{\Gamma}$, and write all ID's except the first in terms of $\tilde{\Gamma}$, i.e., as strings

$$w_0 \# \tilde{w}_1^R \# \tilde{w}_2 \# \tilde{w}_3^R \# \cdots \# \tilde{w}_{2k} \# \tilde{w}_{2k+1}^R,$$

or

$$w_0 \# \tilde{w}_1^R \# \tilde{w}_2 \# \tilde{w}_3^R \# \cdots \# \tilde{w}_{2k-2} \# \tilde{w}_{2k-1}^R \# \tilde{w}_{2k},$$

where w_{2k+1}^R and w_{2k} are proper halting ID. Then, we just have to apply a homomorphism, h , that erases all the symbols not in Σ , and we get

$$L = h(L_1 \cap L_2).$$

Remark. We could also fix the TM so that it outputs $w1$ after it halts with output 1 (in a proper ID). Then, we can write all ID's, except the last, in the special alphabet. This way, the homomorphism will pick out the input, w . But there is a little problem: If the number of moves is odd, we end up with a *reversed* ID! Thus, we also need to make sure that our TM makes an even number of moves. It is not so obvious how to fix the TM to do that and the first solution appears to be simpler.