| *Discrete Structures* | *Prelim 2 sample questions— **Solutions*** |
|---|---|
| *CS2800* | *Questions selected for Spring 2019* |

1. *Prove that $7^m - 1$ is divisible by 6 for all positive integers $m$.*

   **Solution** There are two ways to do this. One way: notice that $7 \equiv 1 \mod 6$, thus $7^m \equiv 1 \mod 6$ for any $m$ (applying the known result that "if $a \equiv b \pmod m$ and $c \equiv d \pmod m$ then $ac \equiv bd \pmod m$" $m-1$ times), and thus $7^m - 1 \equiv 0 \mod 6$. This implies $7^m - 1$ is divisible by 6.

   Alternatively you can do a direct proof by induction:

   **Base case:** $m = 1$, $7^1 - 1 = 6$ which is obviously divisible by 6.

   **Inductive step:** Assume $7^m - 1$ is divisible by 6 for some $m \geq 1$ (inductive hypothesis). Then $7^{m+1} - 1 = 7^{m+1} - 7 + 6 = 7(7^m - 1) + 6$. But $7^m - 1$ is divisible by 6 (by the inductive hypothesis) and so is 6, so $7^{m+1} - 1$ is also divisible by 6. Hence proved by induction.

2. *Suppose that Alice sends the message a to Bob, encrypted using RSA. Suppose that Bob's implementation of RSA is buggy, and computes $k^{-1} \mod 4\phi(m)$ instead of $k^{-1} \mod \phi(m)$. What decrypted message does Bob see? Justify your answer.*

   **Solution** Alice transmits $a^k \mod m$ to Bob, who then computes $(a^k)^{k^{-1}} \mod m$. Because Bob miscomputed $k^{-1}$, we know that $kk^{-1} \equiv 1 \mod 4\phi(m)$. In other words, $kk^{-1} = 1 + t \cdot 4\phi(m)$ for some $t$. Therefore Bob receives

$$\begin{aligned}(a^k)^{k^{-1}} &\equiv a^{1+4t\phi(m)} \\ &\equiv a \cdot a^{4t\phi(m)} \\ &\equiv a \cdot (a^{\phi(m)})^{4t} \\ &\equiv a \cdot 1^{4t} \\ &\equiv a \mod m\end{aligned}$$

3. *(a) What are the units of $\mathbb{Z}$ mod 12?*

   **Solution** A unit in a set of numbers is a number that has an inverse. In the set $\mathbb{Z}_{12} = \{[0], [1], [2], [3], [4], [5], [6], [7], [8]$ the units are [1], [5], [7], and [11]. In general, $[n]$ is a unit mod $m$ if $n$ and $m$ are relatively prime.

   *(b) What are their inverses?*

   **Solution** $[1]^{-1} = [1]$, $[5]^{-1} = [5]$, $[7]^{-1} = [7]$, and $[11]^{-1} = [11]$. This is because $[1] \cdot [1] = [1]$, $[5] \cdot [5] = [25] = [1]$, $[7] \cdot [7] = [49] = [1]$ and $[11] \cdot [11] = [121] = 1$.

   *(c) What is $\phi(12)$?*

   **Solution** By definition of $\phi$, $\phi(12)$ is the number of units mod 12. Since there are 4 units, $\phi(12) = 4$.

4. *Use Euler's theorem and repeated squaring to efficiently compute $8^n \mod 15$ for $n = 5$, $n = 81$ and $n = 16023$. Hint: you can solve this problem with 4 multiplications of single digit numbers. Please fully evaluate all expressions for this question (e.g. write 15 instead of $3 \cdot 5$).*

**Solution** We use the fact that $8^{\varphi(15)} = 1 \mod 15$. $\varphi(15) = (3-1)(5-1) = 8$ [multiplication #1], so we can reduce all of the exponents mod 8. We then use repeated squaring to compute $8^{2^k}$:

$$\begin{aligned} [8]^1 &= [8] \\ [8]^2 &= [64] = [4] && \text{[multiplication \#2]} \\ [8]^4 &= [4]^2 = [16] = [1] && \text{[multiplication \#3]} \end{aligned}$$

We can then use these to compute the powers of [8]:

$$\begin{aligned} [8]^5 &= [8]^4[8] = [1][8] = [8] \\ [8]^{81} &= [8]^1 = [8] \\ [8]^{16023} &= [8]^7 = [8]^4[8]^2[8] = [1][4][8] = [32] = [2] && \text{[multiplication \#4]} \end{aligned}$$

5. *In this problem, we are working mod 7, i.e. $\equiv$ denotes congruence mod 7 and $[a]$ is the equivalence of $a$ mod 7.*

   (a) *What are the units of $\mathbb{Z}_7$? What are their inverses?*

   **Solution**

   - [1]'s inverse is [1]
   - [2]'s inverse is [4]
   - [3]'s inverse is [5]
   - [4]'s inverse is [2]
   - [5]'s inverse is [3]
   - [6]'s inverse is [6]

   (b) *Compute $[2]^{393}$.*

   **Solution** $[2]^{393} = ([2]^3)^{131} = [1]^{131} = [1]$

6. (a) *Recall Bézout's identity from the homework: for any integers $n$ and $m$, there exist integers $s$ and $t$ such that $gcd(n, m) = sn + tm$. Use this to show that if $gcd(k, m) = 1$ then $[k]$ is a unit of $\mathbb{Z}_m$.*

   **Solution** If $gcd(k, m) = 1$ then $1 = sk + tm$. Reducing this equation mod $m$ gives $[1] = [s][k] + [t][0] = [s][k]$. Therefore, $[k]$ has an inverse (namely $[s]$); and is thus a unit.

   (b) *Use part (a) to show that if $p$ is prime, then $\phi(p) = p - 1$.*

   **Solution** Since $p$ is prime, everything less than $p$ is relatively prime to $p$, except for 0. There are $p - 1$ such numbers, and thus $p - 1$ units.

   (c) *Use Euler's theorem to compute $3^{38} \mod 37$ (note: 37 is prime).*

**Solution** $\varphi(37) = 36$, so $[3]^{38} = [3]^2 = [9] \bmod 37$.

7. *Bob the Bomber wishes to receive encrypted messages from Alice the Accomplice. He generates a public key pair $m = 21$ and $k = 5$. Luckily, you have access to an NSA supercomputer that was able to factor 21 into $7 \cdot 3$.*

   (a) *Use this information to find the decryption key $k^{-1}$.*

   **Solution** We must find the inverse of 5 mod $\phi(m) = \phi(7 \cdot 3) = (7-1)(3-1) = 12$. Experimentally, $[5 \cdot 5] = [25] = [1]$. Alternatively, you can use the pulverizer. This results in $1 = -2 \cdot 12 + 5 \cdot 5$, giving an inverse of 5.

   (b) *Without changing $m$, what other possible keys $k$ could Bob have chosen? Find the decryption keys for those keys as well.*

   **Solution** By inspection, the units of $\mathbb{Z}_{12}$ are [1], [5], [7], and [11] (all other numbers share a factor with 12). Experimentally, they are all their own inverses. Note that [1] is not a smart key choice, but we accepted it.

   (c) *Alice encrypts a secret message msg using Bob's public key ($k = 5$), and sends the ciphertext $c = 4$. What was the original message?*

   **Solution** We must compute $[4]^{[5]} = [4^5]$. We see $[4^2] = [16]$; squaring this gives $[4^4] = [(4^2)^2] = [256] = [4]_{21}$. Thus $[4^5] = [4 \cdot 4^4] = [16]$.

8. *Which of the following does RSA depend on? Explain your answer briefly.*

   (a) *Factoring is easy and testing primality is hard.*

   (b) *Factoring is hard and testing primality is easy.*

   (c) *Both factoring and testing primality are hard.*

   (d) *Both factoring and testing primality are easy.*

   **Solution** RSA depends on (b), that factoring is hard and testing primality is easy. The public key in RSA is the product of two large primes. We couldn't generate public keys if testing primality wasn't easy. On the other hand, we could easily decrypt encrypted messages if factoring was easy (because in that case, given a public key, which is the product of two large primes could easily compute the secret – the two factors, and that's what we need to know to decrypt.

9. (a) *Let $m$ and $n$ be integers greater than 1. Show that the function $f : \mathbb{Z}_m \times \mathbb{Z}_n \to \mathbb{Z}_m$ given by $f : ([a]_m, [b]_n) \to [a + b]_m$ is not necessarily well defined. [Hint: you just need an example here.]*

   **Solution** Consider $m = 2$, $n = 3$. Then $[1]_m = [3]_m$ and $[1]_n = [4]_n$, but $[1 + 3]_m = [0]_m$ while $[3 + 4]_m = [1]_m$.

   (b) *Show that $f$ is well defined if $m|n$.*

   **Solution** Suppose $m|n$. Then $n = mc$ for some $c$. Suppose also that $[a]_m = [a']_m$ (so that $a = a' + md$ for some $d$) and $[b]_n = [b']_n$ (so that $b = b' + ne$).

   Then

   $$a + b = (a' + md) + (b' + ne) = a' + b' + md + mce = a' + b' + m(d + ce)$$

   Thus $[a + b]_m = [a' + b']_m$.

10. *We define a set $S$ of functions from $\mathbb{Z}$ to $\mathbb{Z}$ inductively as follows:*

    **Rule 1.** *For any $n \in \mathbb{Z}$, the translation (or offset) function $t_n : x \mapsto x + n$ is in $S$.*

    **Rule 2.** *For any $k \neq 0 \in \mathbb{Z}$, the scaling function $r_k : x \mapsto kx$ is in $S$.*

    **Rule 3.** *If $f$ and $g$ are elements of $S$, then the composition $f \circ g \in S$.*

    **Rule 4.** *If $f \in S$ and $f$ has a right inverse $g$, then $g$ is also in $S$.*

    *In other words, $S$ consists of functions that translate and scale integers, and compositions and right inverses thereof.*

    > ***Note:*** *This semester, we made a bigger distinction between the elements of an inductively defined set and the meaning of an inductively defined set. We probably would have phrased this question as follows: Let $S$ be given by*
    >
    > $$s \in S ::= t_n \mid r_k \mid s_1 \circ s_2 \mid rinv\ s$$
    >
    > *and inductively, let the function defined by $s$ (written $F_s : \mathbb{Z} \to \mathbb{Z}$) be given by the rules $F_{t_n}(x) ::= x + n$, $F_{r_k}(x) ::= ks$, $F_{s_1 \circ s_2}(x) ::= F_{s_1} \circ F_{s_2}$ and let $F_{rinv\ s} ::= g$ where $g$ is a right inverse of $F_s$.*

    (a) *[1 point] Show that the function $f : x \mapsto 3x + 17$ is in $S$.*

    **Solution** By rule 1, the function $t_{17} : x \mapsto x + 17$ is in $S$, and by rule 2, $r_3 : x \mapsto 3x$ is in $S$. By rule 3, therefore, $t_{17} \circ r_3 : x \mapsto 3x + 17$ is in $S$.

    (b) *Use structural induction to prove that for all $f \in S$, $f$ is injective. You may use without proof the fact that the composition of injective functions is injective.*

    **Solution** We must show that all functions formed with each of the rules are injective. Let $P(s)$ be the statement $s$ is injective.

    $P(t_k)$ holds, because $t_k$ has a two sided inverse $t_{-k}$, and is therefore injective.

    $P(r_k)$ holds, because we required that $k \neq 0$. Therefore, if $kx_1 = kx_2$, we can cancel $k$ to find $x_1 = x_2$.

    $P(f \circ g)$ holds, assuming $P(f)$ and $P(g)$, because the composition of injections is an injection.

    If $g$ is the right inverse of $f$, then $P(g)$ holds, because $g$ has a left-inverse (namely $f$) and is therefore injective.

    (c) *Give a surjection $\phi$ from $S$ to $\mathbb{Z}$ (proof of surjectivity not necessary). Remember that this surjection must map a* function *to an* integer*, and for every integer there must be a function that maps to it.*

    **Solution** Let $\phi(s) ::= s(0)$. This is a surjection, because $t_n(0) = 0 + n = n$, so for any $n$ there exists $s \in S$ (namely $t_n$) with $\phi(s) = n$.

11. *Given DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$, we can construct a machine $M_{12}$ with $L(M_{12}) = L(M_1) \cap L(M_2)$ as follows:*

    - *Let $Q = Q_1 \times Q_2 =$ the set of all ordered pairs $(q_1, q_2)$, where $q_1 \in Q_1$ and $q_2 \in Q_2$.*
    - *Let $q_0 \in Q = (q_{01}, q_{02})$.*
    - *Let $F = F_1 \times F_2 = \{(q_1, q_2) \mid q_1 \in F_1$ and $q_2 \in F_2\}$.*
    - *Let $\delta_{12}((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$.*
    - *Let $M_{12} = (Q, \Sigma, \delta_{12}, q_0, F)$.*

*Use structural induction to prove that for all* $x \in \Sigma^*$, $\widehat{\delta}_{12}((q_1, q_2), x) = \left(\widehat{\delta}_1(q_1, x), \widehat{\delta}_2(q_2, x)\right)$.

**Solution** The main challenge here is wading through the notational jungle to understand what the problem actually says. Once you've done this, the proof is short and straightforward. Here it is in all its glory:

We will prove the result by structural induction on $x$, as suggested. Both the set of strings $\Sigma^*$ and the extended transition function $\widehat{\delta}$ are defined recursively (see the definitions at the end). The base case for $x$ is the empty string $\epsilon$. We can simply read off the corresponding line in the definition of $\widehat{\delta}$, which tells us that

- $\widehat{\delta}_1(q_1, \epsilon) = q_1$,
- $\widehat{\delta}_2(q_2, \epsilon) = q_2$, and
- $\widehat{\delta}_{12}((q_1, q_2), \epsilon) = (q_1, q_2)$.

Hence $\widehat{\delta}_{12}((q_1, q_2), \epsilon) = (q_1, q_2) = \left(\widehat{\delta}_1(q_1, \epsilon), \widehat{\delta}_2(q_2, \epsilon)\right)$, so the statement is true in the base case.

Now assume the statement is true for some string $x$, and consider the "next larger" string $xa$.

Again reading off the appropriate line in the definition of $\widehat{\delta}$, we know that

- $\widehat{\delta}_1(q_1, xa) = \delta_1(\widehat{\delta}_1(q_1, x), a)$, and
- $\widehat{\delta}_2(q_2, xa) = \delta_2(\widehat{\delta}_2(q_2, x), a)$

What is $\widehat{\delta}_{12}((q_1, q_2), xa)$? Well, we also have

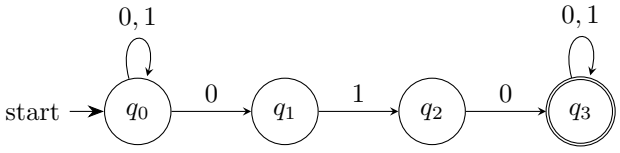
$$\begin{aligned}
\widehat{\delta}_{12}((q_1, q_2), xa) &= \delta_{12}(\widehat{\delta}_{12}((q_1, q_2), x), a) && \text{(definition of } \widehat{\delta}_{12}) \\
&= \delta_{12}\left(\left(\widehat{\delta}_1(q_1, x), \widehat{\delta}_2(q_2, x)\right), a\right) && \text{(inductive hypothesis)} \\
&= \left(\delta_1\left(\widehat{\delta}_1(q_1, x), a\right), \delta_2\left(\widehat{\delta}_2(q_2, x), a\right)\right) && \text{(definition of } \delta_{12}) \\
&= \left(\widehat{\delta}_1(q_1, xa), \widehat{\delta}_2(q_2, xa)\right) && \text{(definition of } \widehat{\delta}_1, \widehat{\delta}_2)
\end{aligned}$$

This proves the statement for all strings $x \in \Sigma^*$ by (structural) induction.

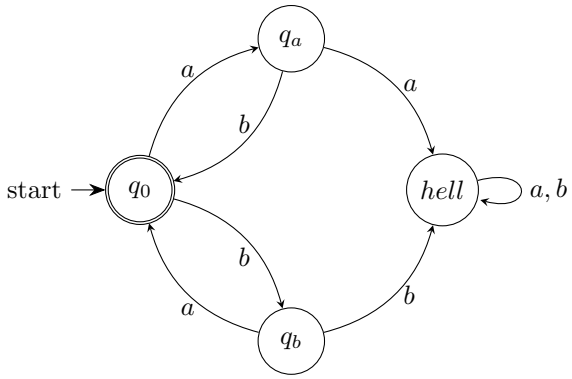
*12. Draw a finite automaton (DFA, NFA or $\epsilon$-NFA) with alphabet $\{0, 1\}$ to recognize the language*

$$\{x \in \{0, 1\}^* \mid x \text{ contains the substring } 010\}$$

**Solution** An NFA is probably the easiest to construct.



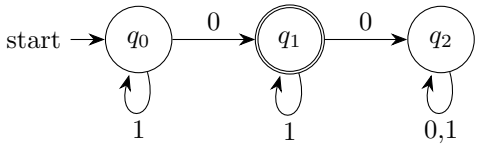

*13. Draw a finite automaton (DFA, NFA or $\epsilon$-NFA) with alphabet $\{a, b\}$ to recognize strings of the form $x_1x_2x_3\cdots$ where each $x_i$ is either "ab" or "ba".*




**Solution**

*14. Build a deterministic finite automaton that recognizes the set of strings of 0's and 1's, that only contain a single 0 (and any number of 1's). Describe the set of strings that lead to each state.*

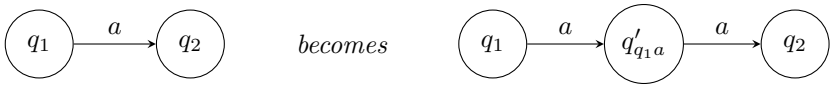
**Solution**




The strings leading to $q_i$ contain $i$ 0's (2 or more for $q_2$).

*15. Given a string $x$, we can define the "character doubling" of $x$ to be $x$ with every character doubled: for example $cd(abc) = aabbcc$. Formally, $cd(\epsilon) = \epsilon$, and $cd(xa) = cd(x)aa$. We can then define the "character doubling" of a language $L$ to be the set of all strings formed by doubling the characters of strings in $L$; formally $cd(L) = \{cd(x) \mid x \in L\}$.*

*Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we can construct a new DFA $M_{cd}$ that recognizes $cd(L(M))$ by adding a new state $q'_{qa}$ to the middle of every transition from $q$ on character $a$:*




*(a) Formally describe the components $(Q_{cd}, \Sigma_{cd}, \delta_{cd}, q_{0cd}, F_{cd})$ of $M_{cd}$ in terms of the components of $M$. Be sure to describe $\delta_{cd}$ on all inputs (you may need to add one or more additional states).*

**Solution** $Q_{cd} = Q \cup \{q'_{qa} \mid q \in Q, a \in A\} \cup \{X\}$

$\delta_{cd} : (q, a) \mapsto q'_{qa}$; $\delta_{cd} : (q'_{qa}, a) \mapsto \delta(q, a)$; $\delta_{cd} : (q'_{qa}, b) \mapsto X$ if $a \neq b$, and $\delta_{cd} : (X, a) \mapsto X$.

Here $X$ is a "Hotel California" state that never accepts. If a character is followed by a different character, then it transitions to $X$.

The remaining components are unchanged: $\Sigma_{cd} = \Sigma$, $q_{0cd} = q_0$, and $F_{cd} = F$.

*(b) Use structural induction on $x$ to prove that for all $x$, $\widehat{\delta}(q_0, x) = \widehat{\delta}_{cd}(q_{0cd}, cd(x))$.*

**Solution** Let $P(x)$ be the statement that $\widehat{\delta}(q_0, x) = \widehat{\delta}_{cd}(q_0, cd(x))$. I will prove $\forall x, P(x)$ by structural induction.

To show $P(\varepsilon)$, note that $\widehat{\delta}(q_0, \epsilon) = q_0$. Moreover, $cd(\varepsilon) = \epsilon$, so $\widehat{\delta}_{cd}(q_0, cd(\varepsilon)) = \widehat{\delta}_{cd}(q_0, \epsilon) = q_0 = \widehat{\delta}(q_0, \varepsilon)$, as required.

To show $P(xa)$, we assume the inductive hypothesis $P(x)$. we compute:

$$
\begin{aligned}
\widehat{\delta}_{cd}(q_0, cd(xa)) &= \widehat{\delta}_{cd}(q_0, cd(x)aa) && \text{by definition of } cd \\
&= \delta_{cd}(\delta_{cd}(\widehat{\delta}_{cd}(q_0, cd(x)), a), a) && \text{by definition of } \widehat{\delta}_{cd} \\
&= \delta_{cd}(\delta_{cd}(\widehat{\delta}(q_0, x), a), a) && \text{by definition of } \widehat{\delta}_{cd} \\
&= \delta_{cd}(q_{(\widehat{\delta}(q_0,x))a}, a) && \text{by definition of } \delta_{cd} \\
&= \delta(\widehat{\delta}(q_0, x), a) && \text{by definition of } \delta_{cd} \\
&= \widehat{\delta}(q_0, xa) && \text{by definition of } \widehat{\delta}
\end{aligned}
$$

(c) *We can also define the "string doubling" of $x$ to be $xx$. For example, $sd(abc) = abcabc$. Show that the set of regular languages is* not *closed under string doubling. In other words, give a regular language $L$ and prove that $sd(L) = \{sd(x) \mid x \in L\}$ is not regular.*

*You can use any theorem proved in class to help prove this result.*

**Solution** Let $L = 0^*1$. Clearly $L$ is regular. Moreover, $sd(L) = \{0^n10^n1 \mid n \in \mathbb{N}\}$.

This language is not regular. To see this, assume for the sake of contradiction that it is. Then there exists some natural number $m$ as in the pumping lemma. Let $x = 0^m10^m1$. Clearly $x \in sd(L)$, and $|x| \geq m$, so we can split $x$ into $u$, $v$, and $w$, as in the pumping lemma. We know that $|uv| \leq n$, so $v$ can only contain 0's. Then $x' = uv^2w$ contains more 0's before the first 1 than after, and thus $x' \notin sd(L)$. But the pumping lemma says that $x' \in sd(L)$; this is a contradiction, and thus $sd(L)$ is not regular.

16. *Happy Cat has been shown the following proof, and has promptly turned into Grumpy Cat. Briefly but clearly identify the error which has induced grumpiness.*

To prove: *The language of the regular expression $0^*1^*$ is, in fact, not DFA-recognizable.*

Proof. *Let $L$ be the language of $0^*1^*$. Assume there is some DFA $M$ with $n$ states that recognizes $L$. Let $x = 0^{n-1}11$. Clearly, $x \in L$ and $|x| \geq n$. Therefore according to the Pumping Lemma, we can split $x$ into three parts $u$, $v$ and $w$, such that $|uv| \leq n$, $|v| \geq 1$, and $uv^iw \in L$ for all natural numbers $i$. Let $|v| = n$. Since $|uv| \leq n$, it must be the case that $u = \epsilon$, and $v = 0^{n-1}1$. Then $uv^2w = 0^{n-1}10^{n-1}11$, which is clearly not in $L$. This contradicts our assumption that there is a DFA which recognizes the language.*

**Solution** The pumping lemma says there exists some $v$, but we have chosen a specific $v$. It may be that the pumping lemma gives some other $v$ (such as 0).

*Prove that $L = \{0^n1^m \mid m \geq n^2\}$ is not DFA-recognizable.*

**Solution** Suppose for the sake of contradiction that $L$ is recognizable. Then there exists $n \in \mathbb{N}$ as in the pumping lemma. Let $x = 0^n1^{n^2}$. Clearly $x \in L$, and clearly $len(x) \geq n$, so there exist $u$, $v$, and $w$, as in the pumping lemma.

Now, $v$ can contain only 0's, since it falls within the first $n$ characters of $x$. Therefore, $uvvw$ has at least $(n + 1)$ 0's, and at most $n^2$ 1's. This means it is not in $L$. But the pumping lemma says $uvvw \in L$, a contradiction.

17. (a) *In lecture, we proved that if $[a]_m$ is a unit, then $[a]_m^{\varphi(m)} = [1]$.*

*Use this to show that $a^{[b]_{\varphi(m)}} := [a^b]_m$ is well defined.*

**Solution** Suppose $[b] = [b']$. Then $b = b' + km$ for some $k$. Therefore,

$$\begin{aligned}
[a^b]_m &= [a^{b'+k\varphi(m)}]_m && \text{plugging in } b = b' + km \\
&= [a^{b'}(a^{\varphi(m)})^k] && \text{algebra} \\
&= [a^{b'}]([a]^{\varphi(m)})^k && \text{definition of modular multiplication} \\
&= [a^{b'}][1]^k && \text{by assumption} \\
&= [a^{b'}] && \text{algebra}
\end{aligned}$$

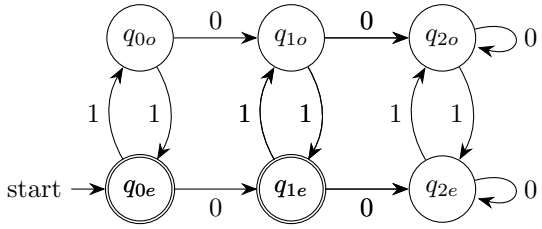
*(b) Use Euler's theorem to prove that if $p$ is prime, then $[a]_p^p = [a]_p$ (whether $[a]_p$ is a unit or not).*

**Solution** Since $p$ is prime, we have $\varphi(p) = p - 1$. If $[a]$ is a unit, then Euler's theorem says that $[a]^{p-1} = [1]$; multiplying both sides by $[a]$ gives $[a]^p = [a]$.

If $[a]$ is not a unit, then $gcd(a, p) \neq 1$; this means that $a$ and $p$ share a common factor, which can only happen if $a$ is a multiple of $p$. In this case, $[a]_p = [0]$, and $[0]^p = [0]$.

*18. Give a DFA that accepts strings in $\{0,1\}^*$ if and only if they contain at most one 0* ***and*** *an even number of 1's. For each state, describe the strings that reach that state.*

**Solution**




If $\hat{\delta}(q_{e0}, x) = q_{0e}$ then $x$ has no 0's and an even number of 1's.
If $\hat{\delta}(q_{e0}, x) = q_{0o}$ then $x$ has no 0's and an odd number of 1's.
If $\hat{\delta}(q_{e0}, x) = q_{1e}$ then $x$ has one 0 and an even number of 1's.
If $\hat{\delta}(q_{e0}, x) = q_{1o}$ then $x$ has one 0 and an odd number of 1's.
If $\hat{\delta}(q_{e0}, x) = q_{2e}$ then $x$ has two or more 0's and an even number of 1's.
If $\hat{\delta}(q_{e0}, x) = q_{2o}$ then $x$ has two or more 0's and an odd number of 1's.

*19. In this question we formalize the usual algorithm for adding numbers represented in base $b$. Let $\Sigma = \{0, 1, \ldots, b-1\}$.*

*(a) Give an* inductive *definition of the base $b$ interpretation function $n : \Sigma^* \to \mathbb{N}$. Check that if $b = 2$ then $n(110) = 6$.*

**Solution** $n(\varepsilon) := 0$ and $n(xa) := bn(x) + a$. If $b = 2$ we have

$$\begin{aligned}
n(101) &= 2n(11) + 0 \\
&= 2(2n(1) + 1) \\
&= 2(2(2n(\varepsilon) + 1) + 1) \\
&= 2(2(0 + 1) + 1) \\
&= 2(3) = 6
\end{aligned}$$

*(b) Consider the following inductive definition of a function*

$$add : \Sigma^* \times \Sigma^* \times \{0,1\} \to \Sigma^*$$

*given by*

$$\begin{aligned}
add(\varepsilon, \varepsilon, c) &:= \varepsilon c \\
add(xd, \varepsilon, c) &:= add(x, \varepsilon, q)r \qquad \text{where } q = quot(d + c, b) \text{ and } r = rem(d + c, b) \\
add(\varepsilon, xd, c) &:= add(x, \varepsilon, q)r \qquad \text{where } q = quot(d + c, b) \text{ and } r = rem(d + c, b) \\
add(xd, ye, c) &:= add(x, y, q)r \qquad \text{where } q = quot(d + e + c, b) \text{ and } r = rem(d + e + c, b)
\end{aligned}$$

*Note: you know this algorithm well; c stands for "carry".*

*Prove that $n(add(x, y, c)) = n(x) + n(y) + c$. If multiple cases are substantially similar, you may say so instead of repeating the proof.*

**Solution** Let $P(x, y)$ be the statement $n(add(x, y, c)) = n(x) + n(y) + c$.

We must show $P(\varepsilon, \varepsilon)$, $P(\varepsilon, xa)$, $P(xa, \varepsilon)$ and $P(xa, yb)$.

In the $P(\varepsilon, \varepsilon)$ case, we have $n(add(\varepsilon, \varepsilon, c)) = n(\varepsilon c) = c = n(\varepsilon) + n(\varepsilon) + c$.

In the $P(xa, \varepsilon)$ case, we first assume $P(x, \varepsilon)$. We have

$$\begin{aligned}
n(add(xa, \varepsilon, c)) &= n(add(x, \varepsilon, q))r && \text{with } a + c = qb + r \text{ as in the definition of } add \\
&= bn(add(x, \varepsilon, q) + r && \text{definition of } n \\
&= bn(x) + bn(\varepsilon) + qb + r && \text{by } P(x, \varepsilon) \\
&= bn(x) + 0 + a + c && \text{by definition of quotient and remainder} \\
&= n(xa) + 0 + c && \text{by definition of } n \\
&= n(xa) + n(\varepsilon) + c && \text{by defintion of } n
\end{aligned}$$

The $P(\varepsilon, xa)$ case is identical.

Now, to see $P(xa, yd)$, we first assume $P(x, y)$, $P(xa, y)$ and $P(x, yd)$. We have

$$\begin{aligned}
n(add(xa, yd, c)) &= n(add(x, y, quot(a + d + c, b))rem(a + d + c, b)) && \text{definition of } add \\
&= bn(add(x, y, quot(a + d + c, b))) + rem(a + d + c, b) && \text{definition of } n \\
&= bn(x) + bn(y) + b\ quot(a + d + c, b) + rem(a + d + c, b) && \text{by } P(x, y) \\
&= bn(x) + bn(y) + a + d + c && \text{definition of quotient and remainder} \\
&= n(xa) + n(yd) + c && \text{definition of } n
\end{aligned}$$

as required.

*20. In this question, we focus on a subset $A$ of the set of regular expressions, given inductively by*

$$r \in A ::= a \mid r_1 + r_2 \mid r_1 r_2$$

*(a) Give the inductive definition of the function $L : A \to 2^{\Sigma^*}$ that gives the language of a regular expression.*

**Solution**

$$\begin{aligned} L(a) &:= \{\varepsilon a\} \\ L(r_1 r_2) &:= \{x_1 x_2 \mid x_1 \in L(r_1) \text{ and } x_2 \in L(r_2)\} \\ L(r_1 + r_2) &:= L(r_1) \cup L(r_2) \end{aligned}$$

*(b) Let $rev(x)$ denote the reverse of $x$ (for example, $rev(\text{"1011"}) = \text{"1101"}$). For an arbitrary language $L$, let $\overleftarrow{L} := \{rev(x) \mid x \in L\}$. Use **structural induction** to show that for all $r \in A$, $\overleftarrow{L(r)}$ is regular. You may use clearly stated facts about $rev(x)$ without proof, as long as they are true.*

**Solution** Let $P(r)$ be the statement "$\overleftarrow{L(r)}$ is regular. We will prove $P(a)$, $P(r_1 r_2)$, $P(r_1 + r_2)$, assuming $P(r_1)$ and $P(r_2)$ in the latter two cases.

To see $P(a)$, note that $rev(a) = a$, and therefore $\overleftarrow{L(a)} = \overleftarrow{\{a\}} = \{a\} = L(a)$ and is thus regular.

For the remaining two cases, assume $P(r_1)$ and $P(r_2)$, i.e. that there are regular expressions $\overleftarrow{r_1}$ and $\overleftarrow{r_2}$ with $L(\overleftarrow{r_1}) = \overleftarrow{L(r_1)}$ and $L(\overleftarrow{r_2}) = \overleftarrow{L(r_2)}$.

To see $P(r_1 r_2)$, I claim that $L(\overleftarrow{r_2}\overleftarrow{r_1}) = \overleftarrow{L(r_1 r_2)}$. To see this, choose any $x \in L(\overleftarrow{r_2}\overleftarrow{r_1})$. Then $x = cat(x_2, x_1)$ where $x_2 \in L(\overleftarrow{r_2})$ and $x_1 \in L(\overleftarrow{r_1})$. Since $x_2 \in \overleftarrow{L(r_2)}$, we have $x_2 = rev(x_2')$ where $x_2' \in L(r_2)$. Similarly $x_1 = rev(x_1')$ for some $x_1' \in L(r_1)$. Therefore $x = x_2 x_1 = rev(x_2')rev(x_1') = rev(x_1 x_2)$ since $rev(xy) = rev(y)rev(x)$. Thus $x \in \overleftarrow{L(r_1 r_2)}$.

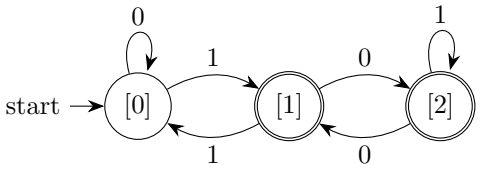
This shows that $L(\overleftarrow{r_2}\overleftarrow{r_1}) \subseteq \overleftarrow{L(r_1 r_2)}$. The reverse inclusion is similar, which gives us that $\overleftarrow{L(r_1 r_2)} = L(\overleftarrow{r_2}\overleftarrow{r_1})$ and is thus regular.

For $P(r_1 + r_2)$, I claim that $L(\overleftarrow{r_1} + \overleftarrow{r_2}) = \overleftarrow{L(r_1 + r_2)}$. To see this, choose an arbitrary $x \in L(\overleftarrow{r_1} + \overleftarrow{r_2})$. Then either $x \in L(\overleftarrow{r_1}) = \overleftarrow{L(r_1)}$ or $x \in \overleftarrow{L(r_2)}$. In either case, $x \in \overleftarrow{L(r_1 + r_2)}$. The reverse inclusion is similar, which gives us that $\overleftarrow{L(r_1 + r_2)} = L(\overleftarrow{r_1} + \overleftarrow{r_2})$, and is thus regular.

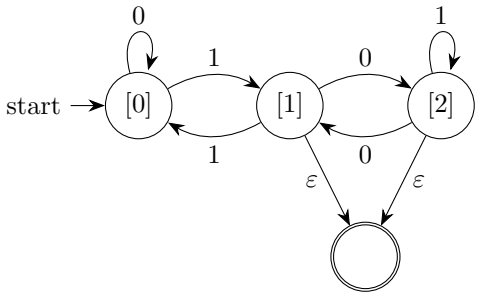
*21. Give a regular expression that matches strings of 0's and 1's that, when interpreted as binary numbers, are* not *divisible by 3.* Hint: *start with an automaton with $Q = \mathbb{Z}_3$.*

Note: *if you clearly indicate your process, partial credit may be given. If you simply write down an answer, credit will be awarded on an all-or-nothing basis.*

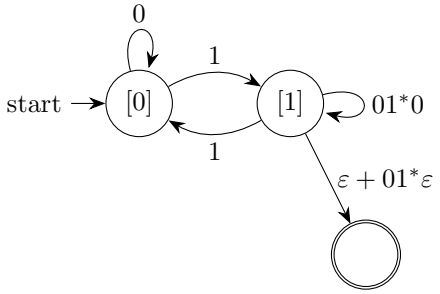
**Solution** As per the hint, we construct an automaton that recognizes the given language.



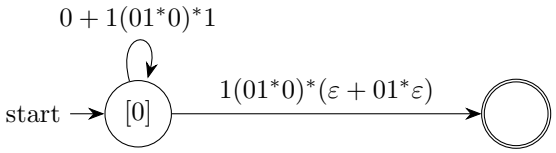

We add a new final state




We remove state [2]:




We remove state [1]:




Which gives us $r = \big(0 + 1(01^*0)^*1\big)^*\big(1(01^*0)^*(\varepsilon + 01^*\varepsilon)\big)$.