Problem Set 7

Due: Start of class on Tuesday, April 15.

Problem 1. Provide recursive definitions of the following sets and functions:

(a) The set $S = \{2^a 3^b 5^c \mid a, b, c \in \mathbb{N}\}$.

Solution. We can define the set S recursively as follows:

1. $1 \in S$
2. If $n \in S$, then $2n$, $3n$, and $5n$ are in S.

(b) The set $L = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid a + 2b = 3k \text{ for some } k \in \mathbb{Z}\}$.

Solution.

1. $(0, 0) \in L$
2. If $(a, b) \in L$, then $(a + 3, b)$, $(a - 3, b)$, $(a + 2, b - 1)$, and $(a - 2, b + 1)$ are in L.

(c) The set Σ^*, which consists of all finite strings of symbols drawn from the set $\{p, n, d, q\}$. (We'll use this set Σ^* in the next three problem parts as well.)

Solution.

1. The empty string λ is in Σ^*.
2. If $\alpha \in \Sigma^*$, then $p\alpha$, $n\alpha$, $d\alpha$, $q\alpha$ are in Σ^*.

(d) Recursively define a function $l : \Sigma^* \to \mathbb{N}$ that maps each string in Σ^* to its length.

Solution.

1. $l(\lambda) = 0$.
2. If $\alpha \in \Sigma^*$, and $\beta \in \{p, n, d, q\}$, then $l(\beta\alpha) = 1 + l(\alpha)$.

(e) Suppose that p stands for penny, n stands for nickel, d stands for dime, and q stands for quarter. Recursively define a function $m : \Sigma^* \to \mathbb{N}$ that maps each string in Σ^* to the monetary value of the corresponding pile of change.

Solution.

1. $m(\lambda) = 0$.
2. If $\alpha \in \Sigma^*$, then $m(p\alpha) = 1 + m(\alpha)$.
3. If $\alpha \in \Sigma^*$, then $m(n\alpha) = 5 + m(\alpha)$.
4. If $\alpha \in \Sigma^*$, then $m(d\alpha) = 10 + m(\alpha)$.
5. If $\alpha \in \Sigma^*$, then $m(q\alpha) = 25 + m(\alpha)$.

(f) Recursively define a function $r : \Sigma^* \to \Sigma^*$ that maps each string $\sigma_1\sigma_2\ldots\sigma_n$ in Σ^* to its reverse:

$$\sigma_n\sigma_{n-1}\ldots\sigma_1$$

Solution.

1. $r(\lambda) = \lambda$.
2. If $\alpha \in \Sigma^*$ and $\beta \in \{p, n, d, q\}$, then $r(\beta\alpha) = r(\alpha)\beta$.

Problem 2. This problem was cancelled due to an error.

Problem 3. Let B be the set of balanced parentheses strings. We want to include all strings of parentheses that might appear in a valid arithmetic expression, such as (()()), ((()))(), and ()((()())()). We want to exclude strings such as)(, (((, and (()())).

(a) Give a recursive definition of B, the set of balanced parentheses strings.

Solution. We define S recursively as follows:

1. $\lambda \in B$
2. If $\alpha \in B$, then $(\alpha) \in B$.
3. If $\alpha, \beta \in B$, then $\alpha\beta \in B$.

(b) Show how ()((()())()) is generated by the rules in your definition.

Solution. We reason as follows:

a.	$\lambda \in B$	rule 1
b.	$() \in B$	a and rule 2
c.	$()() \in B$	b and rule 3
d.	$(()()) \in B$	c and rule 2
e.	$(()())() \in B$	d, b, and rule 3
f.	$((()())()) \in B$	e and rule 2
g.	$()((()())()) \in B$	b, e and rule 3

(c) Give a recursive definition of P, the set of all finite parentheses strings, including those that are not balanced.

Solution. We can define P recursively as follows:

1. $\lambda \in B$
2. If $\alpha \in B$, then $(\alpha \in B$ and $)\alpha \in B$.

(d) Recursively define the functions $L : P \to \mathbb{N}$ and $R : P \to \mathbb{N}$ so that $L[\alpha]$ is the number of left parentheses in the string α and $R[\alpha]$ is the number of right parentheses. (We are using square brackets here, because the argument of L is a string of parentheses!)

Solution. The functions L and R can be defined recursively as follows. For the function L, we have:

1. $L[\lambda] = 0$, $L[(] = 1$, and $L[)] = 0$.
2. For $\alpha, \beta \in P$, $L[\alpha\beta] = L[\alpha] + L[\beta]$.

And for the function R, we have:

1. $R[\lambda] = 0$, $R[(] = 0$, and $R[)] = 1$.
2. For $\alpha, \beta \in P$, $R[\alpha\beta] = R[\alpha] + R[\beta]$.

(e) Use structural induction to prove that $L[\alpha] = R[\alpha]$ for all $\alpha \in B$.

Solution. The proof is by structural induction on the definition of balanced parentheses strings. Let $P(\gamma)$ be the proposition that $L[\gamma] = R[\gamma]$. For the base element λ, we have $L[\lambda] = R[\lambda] = 0$. Next, we show that that $P(\gamma)$ holds for every recursively-generated balanced parentheses string γ, given that it holds for the generating elements. There are two cases to consider.

1. Suppose that $\gamma = (\alpha)$ for some $\alpha \in B$. Then we have:

$$\begin{aligned} L[\gamma] &= L[(\alpha)] \\ &= L[(] + L[\alpha] + L[)] \\ &= L[\alpha] + 1 \\ &= R[\alpha] + 1 \\ &= R[(] + R[\alpha] + R[)] \\ &= R[(\alpha)] \\ &= R[\gamma] \end{aligned}$$

2. Suppose that $\gamma = \alpha\beta$ for some $\alpha, \beta \in B$. Then we can reason as follows:

$$\begin{aligned} L[\gamma] &= L[\alpha\beta] \\ &= L[\alpha] + L[\beta] \\ &= R[\alpha] + R[\beta] \\ &= R[\alpha\beta] \\ &= R[\gamma] \end{aligned}$$

Thus, $L[\gamma] = R[\gamma]$ for every balanced parentheses string γ by structural induction.

(f) Use structural induction to prove that for every prefix α of a balanced parentheses string, we have $L[\alpha] \geq R[\alpha]$.

Solution. The proof uses structural induction on the definition of balanced parentheses strings. Let $P(\gamma)$ be the proposition that for every prefix γ' of the balanced parentheses string γ, we have $L[\gamma'] \geq R[\gamma']$.

First, we show that the proposition holds for the basis element. When γ is the empty string λ, we must have $L[\gamma'] = R[\gamma'] = 0$. Next, we show that the proposition holds for each recursively-generated balanced parentheses string γ, provided it holds for the generating strings. There are two cases.

1. Suppose that $\gamma = (\alpha)$ for some $\alpha \in B$. If the prefix γ' is empty, then $L[\gamma'] = R[\gamma'] = 0$. If γ' is all of γ, then $L[\gamma'] = R[\gamma']$, because $\gamma' = \gamma$ is a balanced parentheses string. Otherwise, γ' consists of (together with a prefix α' of α. Thus, we have:

$$\begin{aligned} L[\gamma'] &= L[(\alpha'] \\ &= L[(] + L[\alpha'] \\ &= 1 + L[\alpha'] \\ &\geq 1 + R[\alpha'] \\ &> R[\alpha'] \\ &= R[(] + R[\alpha'] \\ &= R[(\alpha'] \\ &= R[\gamma'] \end{aligned}$$

2. Otherwise, suppose that $\gamma = \alpha\beta$ for some $\alpha, \beta \in B$. As before, $L[\gamma'] = R[\gamma']$ if γ' is the empty string or all of γ. If γ' is a prefix of α, then the claim holds by induction. If γ' contains all of α and a prefix β' of β, then we have:

$$\begin{aligned} L[\gamma'] &= L[\alpha\beta'] \\ &= L[\alpha] + L[\beta'] \\ &\geq R[\alpha] + R[\beta'] \\ &= R[\gamma'] \end{aligned}$$

Thus, by structural induction, $L[\gamma'] \geq R[\gamma']$ for every prefix of every balanced parentheses string γ

Problem 4. Suppose that you have a regular deck of cards arranged as follows, from top to bottom:

$$A\heartsuit\ 2\heartsuit \ldots K\heartsuit\ A\spadesuit\ 2\spadesuit \ldots K\spadesuit\ A\clubsuit\ 2\clubsuit \ldots K\clubsuit\ A\diamondsuit\ 2\diamondsuit \ldots K\diamondsuit$$

Only two operations on the deck are allowed: *inshuffling* and *outshuffling*. In both, you begin by cutting the deck exactly in half, taking the top half into your right hand and the bottom into your left. Then you shuffle the two halves together so that the cards are perfectly

interlaced; that is, the shuffled deck consists of one card from the left, one from the right, one from the left, one from the right, etc. The top card in the shuffled deck comes from the right hand in an outshuffle and from the left hand in an inshuffle.

(a) Model this problem as a state machine.

Solution. Let the set of states Q be all the possible orderings of the deck of cards. Let the set of start states Q_0 consist of the single ordering listed above. For each state $(c_1, \ldots c_{52}) \in Q$, there are two transitions in δ:

$$\begin{aligned}(c_1, \ldots c_{52}) &\rightarrow (c_1, c_{27}, c_2, c_{28}, \ldots, c_{26}, c_{52}) \\ (c_1, \ldots c_{52}) &\rightarrow (c_{27}, c_1, c_{28}, c_2, \ldots, c_{52}, c_{26})\end{aligned}$$

(b) Use the Invariant Theorem to prove that you can not make the entire first half of the deck black through a sequence of inshuffles and outshuffles.

Solution. The proof uses the Invariant Theorem. Define two cards to be *opposites* if the sum of their positions is 53. For example, the top card is opposite the bottom card, the second card from the top is opposite the second from the bottom, etc.

Let P be the property that $A\heartsuit$ is opposite $K\diamondsuit$, $2\heartsuit$ is opposite $Q\diamondsuit$, etc., as in the initial configuration. We claim that P is an invariant. Suppose that P holds for a given state and consider the two types of transition out of that state. Note that, for both types of transition, cards in opposite positions are mapped to opposite positions. Therefore, the property P still holds. Thus, P is an invariant.

Note that the invariant holds for the start state by definition. However, the invariant can not hold when all black cards are in the top half of the deck; in that case, every black card is opposite a red card. Therefore, by the Invariant Theorem, that state is not reachable.

Note: Discovering a suitable invariant can be difficult! The standard approach is to identify a bunch of reachable states and then look for a pattern, some feature that they all share.[1]

Problem 5. A robot named "Spike" wanders around a two-dimensional grid. He starts out at $(0, 0)$ and is allowed to take four different types of step:

1. $(+2, -1)$
2. $(+1, -2)$
3. $(+1, +1)$
4. $(-3, 0)$

[1] If this does not work, consider twitching and drooling until someone takes the problem away.

Thus, for example, Spike might walk as follows. The types of his steps are listed above the arrows.

$$(0,0) \xrightarrow{1} (2,-1) \xrightarrow{3} (3,0) \xrightarrow{2} (4,-2) \xrightarrow{4} (1,-2) \to \ldots$$

Spike's true love, a perky toaster named "Daisy", awaits at $(0,2)$.

(a) Describe a state machine model of this problem.

Solution. Let the set of states Q be $\mathbb{Z} \times \mathbb{Z}$. Let the set of start states Q_0 consist of the single state $(0,0)$. For each $(x,y) \in Q$, there are four transitions in δ:

$$\begin{aligned}
(x,y) &\to (x+2, y-1) \\
(x,y) &\to (x+1, y-2) \\
(x,y) &\to (x+1, y+1) \\
(x,y) &\to (x-3, y)
\end{aligned}$$

(b) State the most restrictive invariant you can find that holds for all the states that Spike can reach.

Solution. Let P be the property that $x+2y$ is a multiple of 3 for every reachable position (x,y). We show that the property is preserved under each transition in δ. Suppose that $x+2y = 3k$ for some $k \in \mathbb{Z}$. Then we can show that the property still holds after one transition as follows:

$$\begin{aligned}
(x+2)+2(y-1) &= x+2y \\
&= 3k \\
(x+1)+2(y-2) &= x+2y-3 \\
&= 3(k-1) \\
(x+1)+2(y+1) &= x+2y+3 \\
&= 3(k+1) \\
(x-3)+2y &= x+2y-3 \\
&= 3(k-1)
\end{aligned}$$

Therefore, P is an invariant.

(c) Will Spike ever find his true love? Either find a path from Spike to Daisy or use the Invariant Theorem to prove that no such path exists.

Solution. The invariant P holds in the start state, since $0 + 2 \cdot 0 = 0$, which is a multiple of 3. However, the invariant does not hold for Daisy's position, $(0,2)$,

since $0 + 2 \cdot 2 = 4$ is not a multiple of 3. Therefore, by the Invariant Theorem, this is not a reachable state.

Later, though, as it turns out, Spike takes one "illegal" step, sweeps up Daisy, and carries her off to the third quadrant of $\mathbb{C}^2$, where they live happily every after.

Problem 6. The set of *mixup strings* is recursively defined as follows:

1. yxz and zyx are mixup strings.
2. If α is a mixup string, then so is the reverse of α.
3. Below are a set of transformation rules. If the segment on the left side of a rule appears as a substring in a mixup string, then replacing that substring with the segment on the right side of the rule gives a new mixup string. Here, λ denotes the empty string, the string consisting of no symbols.

(**a**)	xy	$\to$	λ
(**b**)	xzz	$\to$	y
(**c**)	y	$\to$	zzz
(**d**)	xz	$\to$	zx
(**e**)	zy	$\to$	xyz
(**f**)	xyx	$\to$	x
(**g**)	z	$\to$	y

Is λ a mixup string? If your answer is yes, prove your answer correct by showing a sequence of valid transformations from one of the base strings to λ. If your answer is no, state an invariant satisfied by all mixup strings, but not satisfied by λ. A full proof is not required.

Solution. The solution is as follows:

xyz	$\to$	xyy	(g)
	$\to$	$xzzzy$	(c)
	$\to$	$xzyzy$	(g)
	$\to$	$xzyyy$	(g)
	$\to$	$xxyzyy$	(e)
	$\to$	$xzyy$	(a)
	$\to$	$xxyzy$	(e)
	$\to$	$xxyxyz$	(e)
	$\to$	$xxyz$	(f)
	$\to$	$xxyy$	(g)
	$\to$	xy	(a)
	$\to$	λ	(a)

This problem was much harder than we intended. You will *not* be expected to wade through such complexities on an exam!

Problem 7. Can you build a $6 \times 6 \times 6$ cube from $1 \times 2 \times 4$ bricks? Justify your answer.

Solution. No. Regard the $6 \times 6 \times 6$ cube as a $3 \times 3 \times 3$ cube of $2 \times 2 \times 2$ subcubes. Color the subcubes black and white in checkerboard style so that the corners are all white. (That is, two subcubes that share a face should be colored oppositely.) Then there are 14 white subcubes and 13 black subcubes.

Now we formulate the construction process as a state machine. The set of states consists of all partial constructions of a $6 \times 6 \times 6$ cube. The start state is is the construction containing no bricks. There is a transition from one state to another if the second can be obtained from the first by adding one brick.

Consider the property that the number of black and white positions covered by bricks are equal. Each brick, no matter how it is oriented, covers an equal number of black and white positions. Therefore, if this property holds for one partial construction, then it holds for every partial construction that can be formed by adding one more brick. Therefore, this property is an invariant.

This invariant holds for the start state, when zero black positions are covered and zero white positions are covered. However, it does not hold for the entire $6 \times 6 \times 6$ block, which has 14×2^3 white positions and 13×2^3 black positions. Therefore, by the Invariant Theorem, a $6 \times 6 \times 6$ cube can not be built.