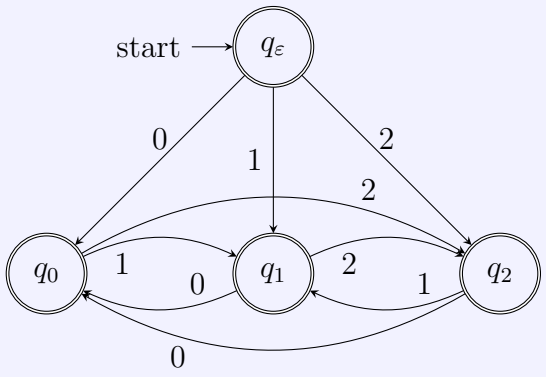
1. **Practice Constructing Automata [3+3+3=9 marks]**
Let $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{2}\}$. Draw an automaton (DFA, NFA or ε-NFA–your choice) for each of the following languages.

(a) $L = \{x \in \Sigma^* \mid x \text{ does not contain } \texttt{00}, \texttt{11}, \text{ or } \texttt{22} \text{ as a substring}\} \subseteq \Sigma^*$.

Solution

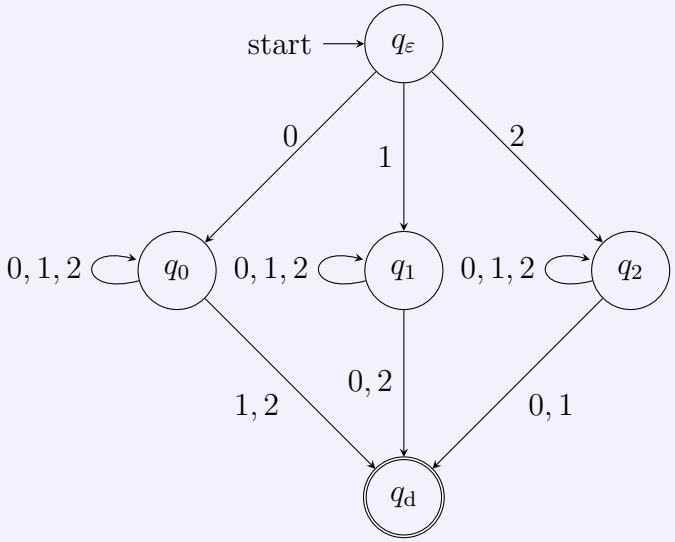
This is either an NFA or a DFA with the dead state omitted (or an ε-NFA with no ε transitions).



(b) $L = \{x \in \Sigma^* \mid \text{the first and last symbol are different } (x_1 \neq x_n)\}$.

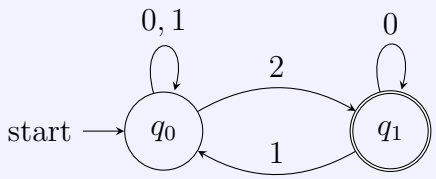
Solution

The DFA has ten states, so we use an NFA.



(c) $L = \{x \in \Sigma^* \mid \text{the last non-}\texttt{0} \text{ symbol in } x \text{ is a } \texttt{2}\}$.

Solution



2. **Unary Regular Languages [2+6=8 marks]**
A *unary language* is any language over an alphabet with one symbol, $\Sigma = \{\mathtt{1}\}$. Consider the following claim.

Claim 1. *Define unary languages $M_{a,b} = \{\mathtt{1}^{a+bi} : i \geq 0\}$ for all non-negative integers a, b. For any unary regular language $L \subseteq \{\mathtt{1}\}^*$ (that is, a language that is both unary and regular), it can be written as a finite union*

$$L = \bigcup_{i=1}^{m} M_{a_i,b_i}$$

where m, $a_1, \ldots, a_m \geq 0$, and $b_1, \ldots, b_m \geq 1$ depend on L.

(a) Unfortunately, there is a small bug in Theorem 1 as written above. Produce a counterexample: a unary regular language that is not of the form in Theorem 1. Briefly justify your counterexample.

Solution

Since $M_{a,b}$ is an infinite language (i.e., contains infinitely many strings) for all $a \geq 0$ and $b \geq 1$, any finite union of the $M_{a,b}$ will also be infinite.[a] Any non-empty finite language is regular, but not a union of $M_{a,b}$'s, and thus a counterexample. E.g., $\{\mathtt{1}\}$ is a counterexample.

[a] Technically, the "empty union" (when $m = 0$) should count as the empty set, $\varnothing$. All *other* unions of $M_{a,b}$ are infinite.

(b) Make a small change to Theorem 1 to give a correct characterization of all unary regular languages. Prove that your characterization is correct.

Solution

All languages in Theorem 1 are regular, but we are missing finite regular languages (as we saw in part (a)). The easiest fix is to allow the b_i's to be zero: take $b_1, \ldots, b_m \geq 0$ in the claim rather than ≥ 1. Note that $M_{a,0} = \{\mathtt{1}^a\}$, so at least any finite unary language can be constructed from those strings.
It remains to prove the repaired claim.

Theorem 2. *For any unary regular language $L \subseteq \{\mathtt{1}\}^*$ (that is, a language*

that is both unary and regular), it can be written as a finite union

$$L = \bigcup_{i=1}^{m} M_{a_i,b_i}$$

where m, $a_1, \ldots, a_m \geq 0$, *and* $b_1, \ldots, b_m \geq 0$ *depend on* L.

Proof. Let L be a unary regular language. There exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizing L, and we will assume WLOG that all states of M are reachable.

Start at q_0 and observe that there is exactly one transition out of each state, $q \to \delta(q, \mathtt{1})$, because the alphabet is unary. For a while, each transition takes us to a new state, which we number sequentially $q_0, q_1, q_2, \ldots$ such that $q_i = \delta(q_{i-1}, \mathtt{1})$ for all $0 \leq i \leq n-1$. Eventually, since the set of states is finite, there is some q_{n-1} such that $q_r = \delta(q_{n-1}, \mathtt{1})$ loops back to an existing state q_r for $0 \leq r \leq n-1$.

By construction, reading $\mathtt{1}^i$ will put the DFA in state q_i for $0 \leq i \leq n-1$. For $i \geq n$, we claim that the DFA transitions to state $q_{i'}$ where $i' = (i - r) \bmod (n - r) + r$. This is because from state q_{n-1} it loops back to $q_r = \delta(q_{n-1}, \mathtt{1})$, and continues around this loop of length $n - r$ forever.

In other words, if we let L_j be the set of strings which take the DFA to state q_j, for all $0 \leq j \leq n-1$, then

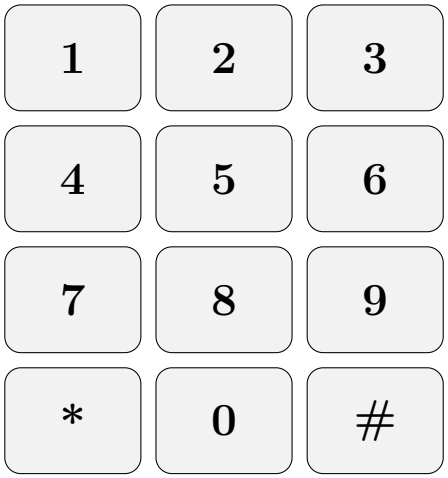
$$L_j = \begin{cases} \{\mathtt{1}^j\} & \text{if } j < r, \\ \{\mathtt{1}^{j+(n-r)m} \mid m \in \mathbb{N}\} & \text{if } j \geq r. \end{cases}$$
$$= \begin{cases} M_{j,0} & \text{if } j < r, \\ M_{j,n-r} & \text{if } j \geq r. \end{cases}$$

Clearly L is the union of L_j for all accepting states q_j. Thus,

$$L = \bigcup_{q_j \in F} L_j = \bigcup_{\substack{j<r \\ q_j \in F}} M_{j,0} \cup \bigcup_{\substack{r \leq j < n \\ q_j \in F}} M_{j,n-r},$$

and L is of the desired form. □

3. **Knight Dialer [4+5=9 marks]**
In a classic interview problem, you are asked to count phone numbers such that each digit is a knight move away from the previous digit (i.e., two steps in one direction and one step in an orthogonal direction) when typed on a pad like the one below. **We also allow repetitions, so adjacent digits can be the same**. Let KNIGHT $\subseteq \Sigma^*$ be the language of all such phone numbers (of any length). E.g., ε, `18816`, and `555` are all in KNIGHT, but `411` and `8675309` are not.



(a) Demonstrate that KNIGHT is a regular language by providing a DFA, and briefly justifying that it recognizes KNIGHT.

Solution

It is a bit annoying to draw this DFA, so we define it mathematically. Let $\Sigma = \{0, 1, \ldots, 9\}$ and let $Q = \{q_{\text{init}}, q_{\text{dead}}\} \cup \Sigma$. We take $q_0 = \bot$ as the initial state, and let $F = Q \backslash \{q_{\text{dead}}\}$ be accepting states. Finally, we define the transition function $\delta \colon Q \times \Sigma \to Q$ by

$$\begin{aligned} \delta(q_{\text{init}}, d) &= d, \\ \delta(q_{\text{dead}}, d) &= q_{\text{dead}}, \\ \delta(q, d) &= \begin{cases} d & \text{if } q = d \text{ or } d \text{ is a knight move from } q, \\ q_{\text{dead}} & \text{otherwise,} \end{cases} \end{aligned}$$

for all $d, q \in \Sigma$.

Why does this recognize KNIGHT? The only outputs of the transition function are either current input symbol, d, or the dead state, q_{dead}, so any non-empty input string will move the automaton to one of those states. If there are two adjacent symbols $d_i d_{i+1}$ in the input which are not either equal or a knight move from each other, then this will lead to the transition $\delta(d_i, d_{i+1}) = q_{\text{dead}}$ to the dead state (if it is not already in the dead state). Hence, the DFA rejects all strings where any step is *not* a knight move or repetition.

On the other hand, $\varepsilon \in$ KNIGHT is clearly accepted by the DFA, and any non-empty string in KNIGHT will cause it to move between the Σ states, and thus eventually accept. All strings in KNIGHT are accepted by the DFA, and only strings in KNIGHT are accepted by the DFA, therefore $\mathcal{L}(M) =$ KNIGHT.

(b) Explain how knowing a DFA for a regular language can help count the number of length-n strings. Apply this to your automaton to count the strings in KNIGHT

of length $n = 7$.

Hint: If we replace all transition labels with $\perp$, then the DFA becomes an NFA which accepts $\perp^n$ if and only if some length n string was originally accepted.

Solution

Recall that to "determinize" an NFA or ε-NFA into a DFA, the new deterministic automaton tracks which states can be reached for a given input. The hint is getting at the idea that tracking whether there *exists* a path to the state is not so different from tracking *how many* paths there are to each state.

Given a general DFA $M = (Q, \Sigma, \delta, q_0, F)$, we define $c(q, \ell)$ to be the number of length-ℓ strings (in Σ^ℓ) that cause the DFA to transition from q_0 to $q \in Q$. It is clear that

$$c(q, 0) = \begin{cases} 1 & \text{if } q = q_0, \\ 0 & \text{if } q \neq q_0. \end{cases} \tag{1}$$

$$c(q, \ell) = \sum_{q' \in Q} c(q', \ell - 1) \cdot |\{a \in \Sigma \mid \delta(q', a) = q\}| \quad \text{for } \ell \geq 1. \tag{2}$$

Let us group the length-ℓ counts into a vector $\mathbf{c}(\ell) = (c(q, \ell))_{q \in Q} \in \mathbb{N}^Q$, where all entries are natural numbers and the dimension is the number of states. We recognize that the recurrence (2) can be expressed as *vector* recurrence: $\mathbf{c}(\ell) = A\,\mathbf{c}(\ell - 1)$ where $A \in \mathbb{N}^{Q \times Q}$ is a $|Q| \times |Q|$ matrix with entries

$$A_{qq'} = |\{a \in \Sigma \mid \delta(q', a) = q\}|$$

for all $q, q' \in Q$. It is clear by induction that $\mathbf{c}(\ell) = A^\ell\,\mathbf{c}(0)$.

Finally, to get the number of *accepting* paths, we need to add up the number of paths for the states in F. We can do this with a vector $\mathbf{f}$ such that

$$\mathbf{f}_q = \begin{cases} 1 & \text{if } q \in F, \\ 0 & \text{otherwise.} \end{cases}$$

Then the total is

$$|\mathcal{L}(M) \cap \Sigma^\ell| = \mathbf{f} \cdot \mathbf{c}(\ell) = \mathbf{f}^\top A^\ell \mathbf{c}(0).$$

In the example of KNIGHT and the automaton from part (a), we have

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 7 & 7 & 7 & 7 & 6 & 9 & 6 & 7 & 7 & 7 & 10 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{c}(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

For length-7 phone numbers, the number we want to calculate is $\mathbf{f}^\top A^7 \mathbf{c}(0)$. Although it is admittedly an annoying calculation to do by hand, it is quite feasible on a computer and exponentially more efficient than, e.g., trying to list all the strings. As it turns out, the answer is 11020.

4. **Pumping Lemma [8 marks]**
Let $\Sigma = \{0, 1\}$ and let $L \subseteq \Sigma^*$ be the language

$$L = \{uv \mid u, v \in \Sigma^*, u = \overline{v}^R\},$$

where $\cdot^R$ denotes string reversal and $\overline{v}$ is bitwise complement, e.g., $\overline{\texttt{01}} = \texttt{10}$. Prove that L is not regular using the pumping lemma.

Solution

Suppose L is regular. Since $B := \{0^a 1^b : a, b \geq 0\} = \{0\}^*\{1\}^*$ is also regular, their intersection, $L \cap B$, is a regular language. Suppose $0^a 1^b \in B$ is also in L and therefore of the form $u\overline{u}^R$. Since u is a prefix, it is of the form $0^i 1^j$ (and either $i = a$ or $j = 0$). However, this makes the full string

$$u\overline{u}^R = 0^i 1^j \overline{0^i 1^j}^R = 0^i 1^j 0^j 1^i,$$

which is not in B unless $i = a = b$ and $j = 0$. We conclude that $0^a 1^b \in L$ if and only if $a = b$. This means $L \cap B = \{0^a 1^a \mid a \geq 0\}$.
We have already shown $\{0^a 1^a \mid a \geq 0\}$ is not regular in class, but let us spell out the pumping lemma argument for practice. Suppose $L \cap B = \{0^a 1^a \mid a \geq 0\}$ is regular and use the pumping lemma to find an integer $p \geq 1$ such that all strings $s \in L \cap B$ of length at least p can be decomposed into xyz such that $|y| > 0$, $|xy| \leq p$, and $xy^i z \in L$ for all $i \geq 0$.
Given this integer p, we consider the string $s = 0^p 1^p$ and note that any decomposition $s = xyz$ with $|xy| \leq p$ will have $y = 0^\ell$ for some $1 \leq \ell \leq p$. Thus, xz will have ℓ fewer zeros (or xy^2z will have ℓ more zeros), but the same number of

1s, so it cannot belong to $L \cap B$ — all strings in $L \cap B$ have the same number of zeros and ones. We conclude that $L \cap B$ is not regular, and therefore L itself is not regular.

5. **Minimization and Reachability [6 marks]**
In class, we saw that we can construct the intersection of two DFAs $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$, $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ with a new automaton $M_{A\cap B}$ (the "product construction") using states $Q_A \times Q_B$. We also saw that $M_{A\cap B}$ is not necessarily minimal. For example,

$$A = \{x \in \{0,1\}^* \mid x \text{ starts and ends with } 0\},$$
$$B = \{x \in \{0,1\}^* \mid x \text{ starts and ends with } 1\},$$

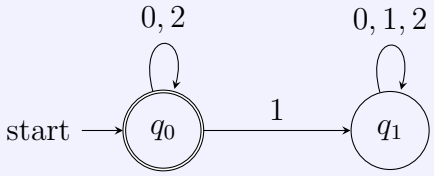
has an accepting state but it is not reachable (clearly, since $A \cap B = \emptyset$).
For this problem, your job is to pick an alphabet Σ and find minimal DFAs M_A and M_B such that every state (in $Q_A \times Q_B$) is reachable, but $M_{A\cap B}$ is still not minimal.

Solution

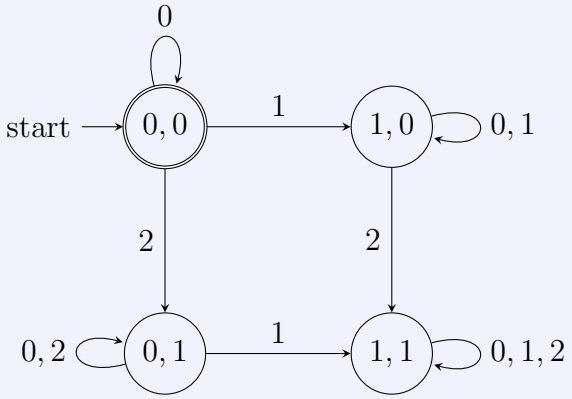
There are many examples, but one of the simplest is to take A, B as follows.

$$A = \{x \in \{0,1,2\} : x \text{ does not contain } 1\},$$
$$B = \{x \in \{0,1,2\} : x \text{ does not contain } 2\}.$$

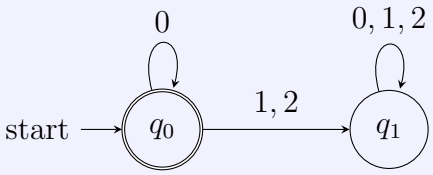
Then A has a two-state finite automaton, shown below. No simpler automaton is possible since with one state we can only accept $\varnothing$ or $\{0,1,2\}^*$.



Similarly, the minimal automaton for B has two states. The intersection of the two is naturally given by a four-state automaton:



All four states are reachable, and yet there is a simpler DFA. Notice that states $(1,0)$, $(0,1)$, and $(1,1)$ have no path to an accepting state, so we can replace them with a single state. Thus, the same language can be accepted by the simpler two-state automaton below.



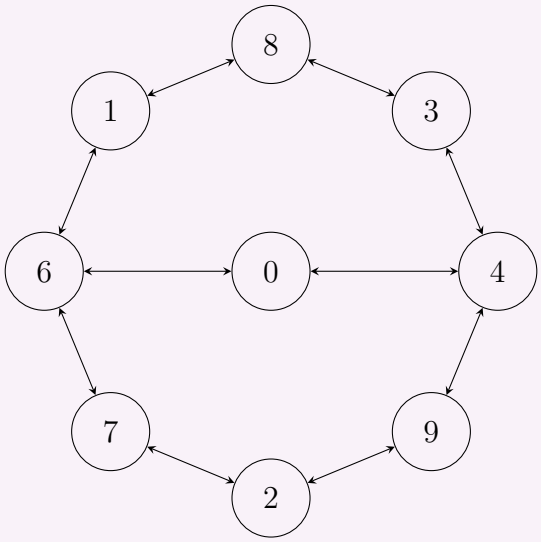
Question 3 Extra

The following is not necessary for full points, nor important for this course, but for the question 3 automaton there are a few tricks which simplify the calculation.

- The number of paths to the dead state (q_{dead}) does not matter because it is not accepting and there is no path out to some state that does accept. We can therefore drop q_{dead} from our vectors and matrices.
- For $\ell > 0$, the number of paths to q_{init} is zero. Like q_{dead}, we can therefore drop coordinate q_{init} from our vectors and matrices.
- $c(5, \ell) = 1$ for all $\ell \geq 1$ because there are no digits a knight's move away from 5, so the only option is to repeat. We can just remember to add 1 for the number $55\cdots5$ at the end, and focus on the other digits.
- By symmetry,

$$\begin{aligned} c(4,\ell) &= c(6,\ell) \\ c(1,\ell) &= c(3,\ell) = c(7,\ell) = c(9,\ell) \\ c(2,\ell) &= c(8,\ell). \end{aligned}$$

  This is easiest to see from the figure.



As a result, we only really need to count $c(0,\ell)$, $c(1,\ell)$, $c(2,\ell)$, $c(4,\ell)$, and then the

equivalent matrices and vectors are as follows:

$$A' = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{pmatrix}, \qquad \mathbf{f}' = \begin{pmatrix} 1 \\ 4 \\ 2 \\ 2 \end{pmatrix}, \qquad \mathbf{c}'(1) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

We get our magic number 11020 from $(\mathbf{f}')^\top (A')^6 \mathbf{c}'(1) + 1$ in this case.
Let $D = \mathrm{diag}([1, 2, 1, 1])$ be the diagonal matrix that doubles the second coordinate.
Define $A'' = D^{-1}AD$, $\mathbf{f}'' = D\mathbf{f}'$ and $\mathbf{c}''(1) = D^{-1}\mathbf{c}(1)$ to get a nicer matrix.

$$A'' = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \qquad \mathbf{f}'' = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 2 \end{pmatrix} \qquad \mathbf{c}''(1) = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \end{pmatrix}.$$

We note that A'' is $I + M$ where

$$M := \begin{pmatrix} 0 & 2F \\ F & 0 \end{pmatrix}, \qquad F := \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

Powers of M have the form

$$M^k = \begin{cases} 2^i \begin{pmatrix} F^k & 0 \\ 0 & F^k \end{pmatrix} & \text{if } k = 2i, \\ 2^i \begin{pmatrix} 0 & 2F^k \\ F^k & 0 \end{pmatrix} & \text{if } k = 2i + 1. \end{cases}$$

And we recognize that powers of F have the form

$$F^i = \begin{pmatrix} F_{i-1} & F_i \\ F_i & F_{i+1} \end{pmatrix},$$

where $F_0 = 0$, $F_1 = 1$, $F_{n+1} = F_n + F_{n-1}$ are the Fibonacci numbers.
Using this, it is a lot of algebra to compute that

$$(\mathbf{f}'')^\top M^{2i} \mathbf{c}''(1) = 2^i(5F_{2i} + 9F_{2i+1})$$
$$(\mathbf{f}'')^\top M^{2i+1} \mathbf{c}''(1) = 2^i(8F_{2i+1} + 12F_{2i+2})$$

It is easy enough to compute the Fibonacci numbers up to $F_7 = 13$, and the following table

k	0	1	2	3	4	5	6
$(\mathbf{f}'')^\top M^k \mathbf{c}''(1)$	9	20	46	104	240	544	1256

Finally, binomial theorem gives

$$(A'')^6 = (I + M)^6 = \sum_{i=0}^{6} \binom{6}{i} M^i,$$

and thus the final answer is

$$1 \cdot (9 + 1256) + 6 \cdot (20 + 544) + 15 \cdot (46 + 240) + 20 \cdot 104 + 1 = 11020.$$