Theory Bridge Exam — Example Questions

Annotated version with some (sometimes rather sketchy) answers and notes.

This is a collection of sample theory bridge exam questions. This is just to get some idea of the format of the bridge exam and the level of difficulty. These example questions do not guarantee anything about the topics of the questions on your bridge exam and they do not guarantee the exact level of difficulty of the questions on your bridge exam.

The actual bridge exam will consist of three or four questions of equal weight.

Procedures of the Theory Bridge Exam

1. The only things that you are allowed to have out during the exam are pen(s), pencil(s), an eraser, and a simple watch. Please note that you are not allowed to use a cell phone, PDA, etc. as a clock.
2. Please silence your cellphones etc.
3. Do not share your supplies with others and do not talk with others.
4. Use only scrap paper provided by us.
5. Please also hand in your scrap paper when you are done.
6. Understanding the questions is part of the exam; you are not allowed to ask questions during the exam.
7. You only have to explain your answer if this is stated in the question.
8. If you use a theorem/construction from your book, please state what you are using where.
9. You may only leave the room after you finish your exam, and if you finish during the last ten minutes of allotted time you must remain seated in the room until time is up.
10. When time is called, you should stop writing immediately, you should remain seated, and you should remain seated without talking until all exams have been collected.
11. Your answers should be to the point.

Notation Used in the Sample Questions

Unfortunately, not every book uses the same notation. Here is some notation used in these sample questions.

- The empty string (sometimes called the null string) is denoted by Λ (you may have seen it denoted by λ, ε, or ϵ).
- A DFA is a deterministic finite automaton, an NFA is a nondeterministic finite automaton without Λ transitions, and an NFA-Λ is a nondeterministic finite automaton that may have Λ transitions.
1. Let $\Sigma = \{0, 1\}$. In this question, every string over $\Sigma$ is viewed as a binary natural number. Leading 0s are allowed. For example, 00100 is viewed as the number 4. $\Lambda$ is viewed as the number 0. Look at the following list of six languages.

- $L_1 = \{ x \in \Sigma^* \mid x \geq 16 \}$.
- $L_2 = \{ x \in \Sigma^* \mid x$ is a square and $x < 1000 \}$.
- $L_3 = \{ x \in \Sigma^* \mid x$ is not divisible by 3 \}.
- $L_4 = \{ x \in \Sigma^* \mid x$ is divisible by 666 \}.
- $L_5 = \{ xy \mid x, y \in \Sigma^* \text{ and } x \ast y = 12 \}$.
  Here, $\ast$ denotes multiplication.
- $L_6 = \{ xy \mid x, y \in \Sigma^*, |x| = |y|$, and $x$ and $y$ represent the same number \}.

(a) Which of the languages from the list are regular?
   $L_1, L_2, L_3, L_4,$ and $L_5$.

(b) Which of the languages from the list contain $\Lambda$?
   $L_2, L_4,$ and $L_6$.

(c) Draw the transition diagram of a DFA that accepts one of the languages from the list or give a regular expression that represents one of the languages from the list. Also state which language you chose.

Different answers are possible. You want to pick a case that is easy and that you sure about. And you should do just one case!

- Regular expression for $L_1$:
  \[ 0^*1(0 + 1)^*(0 + 1)^4 \]

The DFA for $L_1$ is also easy.

- It is fairly easy to come with a DFA for “divisible by 3,” using the fact that an integer is divisible by 3 if and only if the sum of its digits is divisible by 3. Changing the accepting and rejecting states will give a DFA for $L_3$.

- $L_6$ is pretty easy as well. Regular expression:
  \[ 0^*10^*1100 + 0^*100^*110 + 0^*110^*100 + 0^*1000^*11 + 0^*1100^*10 + 0^*11000^*1 \]

- The regular expression for $L_4$ is easy, but long. You need to “+” together all squares less than 1000 in binary and preceed that by $0^*$, since leading 0s are allowed.
2. Let $\Sigma = \{a, b\}$. Let $\text{double}$ be the function from $\Sigma^*$ to $\Sigma^*$ that doubles each character in a string. For example, $\text{double}(baaba) = bbaaaabbbaa$.

(a) What is $\text{double}(aabaa)$?
   $\text{aaaabbaaaa}$

(b) What is $\text{double}(\Lambda)$?
   $\Lambda$

(c) Suppose $x \in \{a, b\}^*$ and the length of $x$ is $k$. What is the length of $\text{double}(x)$?
   $2k$

(d) Give a recursive (inductive) definition of function $\text{double}$ from $\Sigma^*$ to $\Sigma^*$.
   - $\text{double}(\Lambda) = \Lambda$.
   - For all $x \in \Sigma^*$ and $\sigma \in \Sigma$, $\text{double}(x\sigma) = \text{double}(x)\sigma\sigma$.

For $L$ a language over $\Sigma$, define $\text{double}(L)$ as follows:
   $$\text{double}(L) = \{ \text{double}(x) \mid x \in L \}.$$  

(e) Let $A$ be the language $\{ab, bbb, baba\}$. What is $\text{double}(A)$?
   $\{aabb, bbbbbb, bbaabbaa\}$

(f) List all languages $B$ over $\Sigma$ that have the property that $\text{double}(B) = B$.
   $\emptyset$ and $\{\Lambda\}$

(g) For each of the following statements, circle the right answer.
   i. If $L$ is a regular language over $\Sigma$, then $\text{double}(L)$ is regular. True
   ii. If $L$ is a finite language over $\Sigma$, then $\text{double}(L)$ is finite. True
   iii. If $L \subseteq \Sigma^*$ is not regular, then $\text{double}(L)$ is not regular. True
   iv. If $L_1$ and $L_2$ are regular languages over $\Sigma$, then $L_1 \cup L_2$ is regular. True
   v. If $L_1$ and $L_2$ are finite languages over $\Sigma$, then $L_1 \cup L_2$ is finite. True
   vi. If $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ are not regular, then $L_1 \cup L_2$ is not regular. False
3. This question is about the subset construction. Recall that this is the construction that is used to construct a DFA that accepts the same language as a given NFA.

In this question, take $\Sigma = \{a, b, c\}$.

(a) If you literally apply the subset construction to an NFA with $k$ states, how many states will the corresponding FA have? $2^k$

(b) Give a simple example of a minimal NFA such that the FA obtained by literally applying the subset construction is not minimal. (A minimal NFA is an NFA such that no equivalent NFA has fewer states.) For your answer, draw the transition diagram of the NFA, the transition diagram of the FA obtained by applying the subset construction, and briefly argue that the FA is not minimal.

Consider the NFA $M_1$ with one rejecting state and no transitions (you need to draw the transition diagram). Clearly, $M_1$ is minimal, since an NFA cannot have fewer than one state. $L(M_1) = \emptyset$ and there exists a one-state FA that accepts $\emptyset$. But the subset construction gives a two-state FA (you need to draw the transition diagram).

(c) Give a simple example of a minimal NFA such that the FA obtained by literally applying the subset construction is minimal. For your answer, draw the transition diagram of the NFA, the transition diagram of the FA obtained by applying the subset construction, and briefly argue that the FA is minimal.

Consider the NFA $M_2$ with one accepting state and no transitions (draw it). Clearly, $M_2$ is minimal, since an NFA cannot have fewer than one state. $L(M_2) = \{\Lambda\}$. The two-state FA given by the subset construction (draw it) is minimal, since an FA that accepts $\Lambda$ needs at least one accepting state, and an FA that does not accept $\Sigma^*$ needs at least one rejecting state.
4. Suppose that \( P \neq NP \) and that the alphabet is \( \{a, b, c\} \). Draw a Venn diagram that shows the relationships between the following classes of languages:

(a) NP

(b) P

(c) the deterministic context-free languages

(d) the languages that can be accepted by an NFA-\( \Lambda \)

(e) the languages that can be accepted by a nondeterministic Turing machine with 25 tapes

(f) the languages that can be accepted by a Turing machine that halts on all inputs

(g) the languages that can be generated by an unrestricted grammar

(h) the languages that have an algorithmic solution

(i) the languages whose complement is recursively enumerable

(j) the recursive languages

(k) the recursively enumerable languages

(l) the regular languages

Your Venn diagram should show the following relationships

\[
(d) = (l) \subseteq (c) \subseteq (b) \subseteq (a) \subseteq (f) = (h) = (j) \subseteq (e) = (g) = (k)
\]

In addition, \((j) \subseteq (i)\) and \((i) \cap (k) = (j)\).
5. (a) Suppose that the alphabet is \{0, 1\}.

For every class of languages in the table, you have to state whether the class is closed under complementation, concatenation, and intersection.

Write “yes” in a box if the class is closed under the operation, write “no” if the class is not closed under the operation, and write “nobody knows” if nobody knows whether or not the class is closed under the operation.

<table>
<thead>
<tr>
<th>closed under</th>
<th>Complementation</th>
<th>Concatenation</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>nobody knows</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>P</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>The context-free languages</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>The recursively enumerable languages</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>The recursive languages</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

(b) For one of the boxes where you answered “no,” give an explicit example that shows why that particular class is not closed under that particular operation. (Clearly state which box you chose.)

To show that a class is not closed under a particular operation, it suffices to give a counterexample. For example, To show that CFLs are not closed under intersection, argue as follows. Let \( A = \{a^i b^i c^j \mid i, j \geq 0\} \) and let \( B = \{a^i b^j c^i \mid i, j \geq 0\} \). Then \( A \) and \( B \) are CFLs, but \( A \cap B = \{a^i b^i c^i \mid i \geq 0\} \), which is not a CFL.

(c) For one of the boxes where you answered “yes,” informally argue why that particular class is closed under that particular operation. (Clearly state which box you chose.)

All “yes” boxes are totally doable.

For example, to show that recursive languages are closed under complementation, argue as follows. Suppose \( A \) is recursive. Then there is an algorithm for \( A \). If you switch the accepts and rejects in this algorithm, you get an algorithm for the complement of \( A \). So, the complement of \( A \) is recursive.
6. Consider the following grammar $G$:

$$
S \rightarrow \Lambda \mid 0S \mid 10S \mid 0S1 \mid 1S0 \mid S01 \mid S10 \mid 1S \mid S1
$$

(a) Show that this grammar is ambiguous.

You need to give a string and two leftmost derivations or two parse trees for this string. For example, these are two leftmost derivations for the string $1$:

$$
S \Rightarrow 1S \Rightarrow 1
$$

and

$$
S \Rightarrow S1 \Rightarrow 1
$$

At first glance, it may seem that the language generated by $G$ is

$$
L = \{ x \in \{0, 1\}^* \mid n_0(x) \leq n_1(x) \}
$$

However, this is not true.

(b) For both of the following statements, circle “true” is the statement is true, and circle “false” if the statement is false.

i. $L \subseteq L(G)$ true / false

ii. $L(G) \subseteq L$ true / false

(c) Show that $L(G) \neq L$.

Argue that you can’t derive $00111100$.

(d) Give a context-free grammar that generates $L$. (It suffices to give the rules.)

$$
S \rightarrow SS \mid 0S1 \mid 1S0 \mid S1.
$$
7. Let $L = \{a^i b^k c^i d^k \mid i, k \geq 0\}$.

(a) Explain informally why $L$ doesn't seem to be a context-free language.

A PDA that accepts $L$ should push the $a$'s and the $b$'s. Then it needs to compare the $c$'s against the $a$'s. But the $a$'s are at the bottom of the stack. To get to the $a$'s, the $b$'s have to be popped. But then it is impossible to compare the $b$'s against the $d$'s.

(b) Prove that $L$ is not context-free.

Suppose for a contradiction that $L$ is context-free. Let $n$ be the integer from the pumping lemma for context-free languages.

Let $u = a^n b^n c^n d^n$. Then $u \in L$ and $|u| \geq n$. So, by the pumping lemma for context-free languages, there exist strings $v, w, x, y, z$ such that $u = vwxyz$, $|wy| > 0$, $|wxy| \leq n$, and for any $m \geq 0$, $vw^mxy^mz \in L$.

Since $|wxy| \leq n$ and $|wy| \geq 1$, it holds that $wy \in a^+ b^*$ or $wy \in b^+ c^*$ or $wy \in c^+ d^*$ or $wy \in d^+$.

We will show that all these cases, $vxz \notin L$.

i. If $wy \in a^+ b^*$, then $vxz$ contains less than $n$ a's and $n$ c's, so $vxz \notin L$.

ii. If $wy \in b^+ c^*$, then $vxz$ contains less than $n$ b's and $n$ d's, so $vxz \notin L$.

iii. If $wy \in c^+ d^*$, then $vxz$ contains less than $n$ c's and $n$ a's, so $vxz \notin L$.

iv. If $wy \in d^+$, then $vxz$ contains less than $n$ d's and $n$ b's, so $vxz \notin L$.

We have shown that $vxz \notin L$. This is a contradiction.

It follows that the assumption that $L$ is context-free is wrong.

So, we have shown that $L$ is not context-free.
8. Consider the following decision problem \textsc{AllOrNothing}\textsubscript{CFG}:
Given a CFG $G$, is $L(G) \in \{\emptyset, \Sigma^*\}$?
(In other words, given a CFG $G$, is it the case that $G$ generates no strings or all strings?)

You may use the fact that the membership problem for CFGs (Given a CFG $G$ and a string $x$, is $x \in L(G)$?) and the emptiness problem for CFGs (Given a CFG $G$, is $L(G) = \emptyset$?) are decidable.
You may also use the fact that ALL\textsubscript{CFG} (Given a CFG $G$, is $L(G) = \Sigma^*$?) is undecidable.

(a) Show that \textsc{AllOrNothing}\textsubscript{CFG} is undecidable. (Hint: use the undecidability of ALL\textsubscript{CFG}.)

Suppose for a contradiction that \textsc{AllOrNothing}\textsubscript{CFG} is decidable.
Then we have the following algorithm for ALL\textsubscript{CFG}:

\begin{verbatim}
Given a CFG G
if L(G) = \emptyset then reject
else if G \in \textsc{AllOrNothing}\textsubscript{CFG} then accept
else reject
\end{verbatim}

But this contradicts the fact that ALL\textsubscript{CFG} is undecidable.

(b) Show that \textsc{AllOrNothing}\textsubscript{CFG} is in \textsc{coRE}.

The following semi-algorithm shows that the complement of \textsc{AllOrNothing}\textsubscript{CFG} is recursively enumerable, which implies that \textsc{AllOrNothing}\textsubscript{CFG} is in \textsc{coRE}.

\begin{verbatim}
Given a CFG G
if L(G) = \emptyset then reject
else for every string $s$ in $\Sigma^*$
    if $s \not\in L(G)$ then accept
\end{verbatim}

(c) Is \textsc{AllOrNothing}\textsubscript{CFG} in \textsc{RE}? Briefly explain your answer.

No. Suppose for a contradiction that \textsc{AllOrNothing}\textsubscript{CFG} is in \textsc{RE}. We know from part (b) that \textsc{AllOrNothing}\textsubscript{CFG} is in \textsc{coRE}. It follows that \textsc{AllOrNothing}\textsubscript{CFG} is decidable, but this contradicts part (a).
9. (From Martin) A Post machine is similar to a PDA, but with the following differences. It is deterministic, it has a queue instead of a stack, and it doesn’t have a separate input tape (at the start of the computation, the input will be on the queue). Items can be added only to the rear of the queue, and deleted only from the front. Assume that there is a marker $Z_0$ initially on the queue following the input. For example, if the input is $abb$, then at the start of the computation, the contents of the queue are $abbZ_0$, the symbol $a$ is at the front of the queue and the symbol $Z_0$ is at the rear of the queue. A single move of the Post machine depends on the state and the symbol currently at the front of the queue. A move has three components: the resulting state, an indication of whether or not to remove the current symbol from the front of the queue, and what string to add to the rear of the queue.

Construct a Post machine to accept the language \{a^n b^n c^n \mid n \geq 0\}. Draw a transition diagram, explain the notation used in your transition diagram, and give a short explanation of your Post machine, so that we can understand it.

The main idea is to keep cycling through the contents of the queue. For each iteration, delete one $a$, one $b$, and one $c$, and accept if the queue becomes empty. You will need to draw five states, say 1, 2, 3, 4, and 5. 1 is the start state, 5 is the accepting state. You have different options for labeling the arrows. I will use $\sigma/x$, where $\sigma$ is a letter and $x$ a string to denote that the symbol at the front of the queue is $\sigma$, that this symbol will be removed, and that $x$ is the string added to the rear of the queue. Add arrows $a/\Lambda$ from 1 to 2, $a/a$ from 2 to 2, $b/\Lambda$ from 2 to 3, $b/b$ from 3 to 3, $c/\Lambda$ from 3 to 4, $c/c$ from 4 to 4, $Z_0/\Lambda$ from 4 to 1, and $Z_0/\Lambda$ from 1 to 5.
10. Consider the following decision problem. (If this were a question on the real bridge exam, the definitions of clique, independent set, and the problem INDEPENDENT SET would be included.)

\[
IS\text{-}OR\text{-}CLIQUE = \{ (G, k) \mid G \text{ is a graph, } k \text{ is a positive integer, and } G \text{ has a clique of size } k \text{ or an independent set of size } k \}\]

(a) Show that \textit{IS-OR-CLIQUE} is in NP.

On input \(G\) and \(k\), guess \(k\) vertices and accept if and only if these \(k\) vertices form a clique or an independent set.

(b) Give an explicit example that shows that the function \(f\) defined as \(f(G, k) = (G, k)\) is not a polynomial-time many-one reduction from \textit{INDEPENDENT-SET} to \textit{IS-OR-CLIQUE}.

For example, let \(G\) be a clique of size 3 and let \(k = 3\). Then \(G\) does not have an independent set of size 3, but \((G, k) \in IS\text{-}OR\text{-}CLIQUE\).

(c) Show that \textit{IS-OR-CLIQUE} is NP-complete.

We’ll reduce \textit{INDEPENDENT-SET} to \textit{IS-OR-CLIQUE}. Define \(f\) as follows: \(f(G, k) = (H, k + n)\), where \(n\) is the number of vertices of \(G\) and \(H\) is \(G\) plus \(n\) isolated vertices. We need to show that \(f\) is a polynomial-time many-one reduction from \textit{INDEPENDENT-SET} to \textit{IS-OR-CLIQUE}. It is immediate that \(f\) is computable in polynomial time. It is also immediate that if \(G\) has an independent set of size \(k\), then \(H\) has an independent set of size \(k + n\), and thus \((H, k + n) \in IS\text{-}OR\text{-}CLIQUE\). Finally, suppose that \((H, k + n) \in IS\text{-}OR\text{-}CLIQUE\). Since \(H\) has \(2n\) vertices and at least \(n\) of these vertices are isolated vertices, \(H\) does not have a clique of size \(k + n\). It follows that \(H\) has an independent set of size \(k + n\). Since \(H\) consists of \(G\) plus \(n\) isolated vertices, it follows that \(G\) has an independent set of size \(k\).