HW3 Circuits and Logic

CSE20F21

Sample Solutions

Assigned questions

1. Imagine a friend suggests the following argument to you: "The compound proposition

$$(x \to y) \to z$$

is logically equivalent to

$$x \to (y \to z)$$

because I can transform one to the other using the following sequence of logical equivalences like distributivity and associativity and using that $(p \to q) \equiv (\neg p \lor q)$:

$$(x \to y) \to z \equiv \neg(\neg x \lor y) \lor z \equiv \neg x \lor (\neg y \lor z) \equiv x \to (\neg y \lor z) \equiv x \to (y \to z)$$

so conditionals are associative just like conjunctions and disjunctions".

(a) (*Graded for correctness*) Prove to your friend that they made a mistake by giving a truth assignment to the propositional variables where the two compound propositions $(x \to y) \to z$ and $x \to (y \to z)$ have different truth values. Justify your choice by evaluating these compound propositions using the definitions of the logical connectives and include enough intermediate steps so that a student in CSE 20 who may be struggling with the material can still follow along with your reasoning.

Solution: Consider the truth assignment $x = F, y = T, z = F$. Then

$$(x \to y) \to z = (F \to T) \to F = T \to F = F$$

where we plugged in the values and then used the definition of conditional to evaluate. On the other hand,

$$x \to (y \to z) = F \to (T \to F) = F \to F = T$$

by plugging in (as before). Logical equivalence of two compound propositions means that they have the same output value for each possible setting of truth values to the propositional variables in the compound propositions. Since we found an assignment of truth values are to the propositional variables that gives different truth values for the two compound propositions, these two compound propositions are not logically equivalent.

(b) (*Graded for fair effort completeness*) Help your friend find the problem in their argument by pointing out which step(s) were incorrect.

Solution: The problem in the argument was in the second claimed logical equivalence, where $\neg(\neg x \vee y) \vee z$ and $\neg x \vee (\neg y \vee z)$ were claimed to be logically equivalent, which they are not.

(c) (*Graded for fair effort completeness*) Give **three** different compound propositions that are actually logically equivalent to (and not the same as)

$$x \to (y \to z)$$

Justify each one of these logically equivalences either by applying a sequence of logical equivalences or using a truth table. Notice that you can use other logical operators (e.g. $\neg, \vee, \wedge, \oplus, \to, \leftrightarrow$) when constructing your compound propositions.

Bonus; not for credit (do not hand in): How would you translate each of the equivalent compound propositions in English? Does doing so help illustrate why they are equivalent?

Solution:

i. Applying the logical equivalence $p \to q \equiv \neg p \vee q$ to the conditional with hypothesis x and conclusion $y \to z$, we get:

$$\neg x \vee (y \to z)$$

ii. Applying the logical equivalence $p \to q \equiv \neg p \vee q$ to the conditional with hypothesis y and conclusion z, we get:

$$x \to (\neg y \vee z)$$

iii. Combining the previous two, and using associativity, we get:

$$\neg x \vee \neg y \vee z$$

iv. Starting with the previous example and using associativity and De Morgan's Law, we get:

$$\neg x \vee \neg y \vee z \equiv (\neg x \vee \neg y) \vee z \equiv \neg(x \wedge y) \vee z$$

v. Starting with the previous example and using the logical equivalence $p \to q \equiv \neg p \vee q$ with $p = (x \wedge y)$ and $q = z$, we get

$$(x \wedge y) \to z$$

Thus, the compound propositions $\neg x \vee (y \to z)$, $x \to (\neg y \vee z)$, $\neg(x \wedge y) \vee z$, and $(x \wedge y) \to z$ are each logically equivalent to $x \to (y \to z)$.

2. For each part of this question you will use the following input-output definition table with four inputs x_3, x_2, x_1, x_0

x_3	x_2	x_1	x_0	*out*
1	1	1	1	1
1	1	1	0	0
1	1	0	1	0
1	1	0	0	0
1	0	1	1	0
1	0	1	0	1
1	0	0	1	0
1	0	0	0	0
0	1	1	1	0
0	1	1	0	0
0	1	0	1	1
0	1	0	0	0
0	0	1	1	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

(a) (*Graded for fair effort completeness*) Draw a logic circuit that corresponds to this input-output table. We recommend the following steps:

- Draw symbols for the inputs on the left-hand-side and for the output on the right-hand side.
- Construct an expression for *out* using (some of) the inputs x_3, x_2, x_1, x_0 and the logic gates XOR, AND, OR, NOT. *Hint:* are normal forms helpful here? How do you choose which normal form to use?
- Draw and label the gates corresponding to the expression you construct, and connect appropriately with wires.

Solution: Using the DNF approach because there are fewer 1 output rows than 0 output rows, we describe each of the rows that output 1:

$$\begin{aligned}
x_3 \wedge x_2 \wedge x_1 \wedge x_0 &\qquad \text{represents } x_3 = 1, x_2 = 1, x_1 = 1, x_0 = 1 \\
x_3 \wedge \neg x_2 \wedge x_1 \wedge \neg x_0 &\qquad \text{represents } x_3 = 1, x_2 = 0, x_1 = 1, x_0 = 0 \\
\neg x_3 \wedge x_2 \wedge \neg x_1 \wedge x_0 &\qquad \text{represents } x_3 = 0, x_2 = 1, x_1 = 0, x_0 = 1 \\
\neg x_3 \wedge \neg x_2 \wedge \neg x_1 \wedge \neg x_0 &\qquad \text{represents } x_3 = 0, x_2 = 0, x_1 = 0, x_0 = 0
\end{aligned}$$

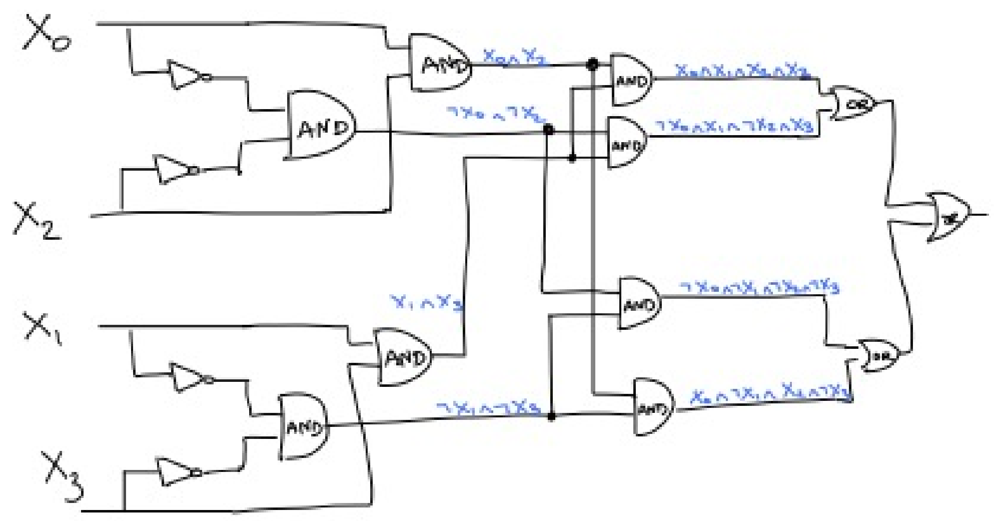
Now, to express that any of these situations leads to output 1, we string together the AND clauses with ORs:

$$(x_3 \wedge x_2 \wedge x_1 \wedge x_0) \vee (x_3 \wedge \neg x_2 \wedge x_1 \wedge \neg x_0) \vee (\neg x_3 \wedge x_2 \wedge \neg x_1 \wedge x_0) \vee (\neg x_3 \wedge \neg x_2 \wedge \neg x_1 \wedge \neg x_0)$$

Circuit diagram: we'll start by using associativity and commutativity to rewrite the compound proposition to facilitate the circuit construction

$$\begin{aligned}
&(\ (x_3 \wedge x_1) \wedge (x_0 \wedge x_2)\ ) \vee \\
&(\ (x_3 \wedge x_1) \wedge (\neg x_0 \wedge \neg x_2)\ ) \vee \\
&(\ (\neg x_3 \wedge \neg x_1) \wedge (x_0 \wedge x_2)\ ) \vee \\
&(\ (\neg x_3 \wedge \neg x_1) \wedge (\neg x_0 \wedge \neg x_2)\ )
\end{aligned}$$

When we draw the diagram, we can reorder the inputs to minimize wire crossing.



(b) (*Graded for correctness*) For each of the following predicates, consider whether this input-output table (and the logic circuit from part (a)) implements the rule for the predicate. If yes, explain why using the definitions of the operations involved and consider **all** possible inputs. If no, explain why not by providing specific example input values and using the input-output definition table to compute the logic circuit's output for this example input and then comparing with the value of the predicate in question to justify your example.

i. $P_1 : \{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\} \to \{T, F\}$ given by

$$P_1(\ (x_3, x_2, x_1, x_0)\ ) = \begin{cases} T & \text{when } (x_3x_2x_1x_0)_{2,4} \leq 5 \\ F & \text{otherwise} \end{cases}$$

Solution: The input-output table $\boxed{\text{does not}}$ implement P_1. We can see why not by considering the example (x_3, x_2, x_1, x_0) with $x_3 = 0$, $x_2 = 0$, $x_1 = 0$, $x_0 = 1$. Using the input-output definition table, the output for this example input is 0. However, using the definition of P_1, we get

$$P_1(\ (0,0,0,1)\ ) = 1$$

because

$$(0001)_{2,4} = 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1$$

which is less than or equal to 5. Since the output from the table doesn't match the predicate definition for this example, the input-output table and circuit do not implement P_1.

ii. $P_2 : \{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\} \to \{T, F\}$ given by

$$P_2(\ (x_3, x_2, x_1, x_0)\ ) = \begin{cases} T & \text{when } (x_3x_2x_1x_0)_{2,4} \textbf{ mod } 5 = 0 \\ F & \text{otherwise} \end{cases}$$

Solution: The input-output table $\boxed{\text{does}}$ implement P_2. To see this, we need to show that the output from the table agrees with P_2 for **all** possible inputs. We can organize the calculations in a table:

x_3	x_2	x_1	x_0	*out*	$P_2(\ (x_3,x_2,x_1,x_0)\ )$, with justification
1	1	1	1	1	1 because $(1111)_{2,4} = 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 15 = 3\cdot 5 + 0$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 3$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 0$
1	1	1	0	0	0 because $(1110)_{2,4} = 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 14 = 2\cdot 5 + 4$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 2$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 4 \neq 0$
1	1	0	1	0	0 because $(1101)_{2,4} = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 13 = 2\cdot 5 + 3$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 2$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 3 \neq 0$
1	1	0	0	0	0 because $(1100)_{2,4} = 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 12 = 2\cdot 5 + 2$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 2$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 2 \neq 0$
1	0	1	1	0	0 because $(1011)_{2,4} = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 11 = 2\cdot 5 + 1$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 2$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 1 \neq 0$
1	0	1	0	1	1 because $(1010)_{2,4} = 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 10 = 2\cdot 5 + 0$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 2$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 0$
1	0	0	1	0	0 because $(1001)_{2,4} = 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 9 = 1\cdot 5 + 4$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 1$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 4 \neq 0$
1	0	0	0	0	0 because $(1000)_{2,4} = 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 8 = 1\cdot 5 + 3$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 1$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 3 \neq 0$
0	1	1	1	0	0 because $(0111)_{2,4} = 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 7 = 1\cdot 5 + 2$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 1$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 2 \neq 0$
0	1	1	0	0	0 because $(0110)_{2,4} = 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 6 = 1\cdot 5 + 1$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 2$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 1 \neq 0$
0	1	0	1	1	1 because $(0101)_{2,4} = 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 5 = 1\cdot 5 + 0$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 1$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 0$
0	1	0	0	0	0 because $(0100)_{2,4} = 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 4 = 0\cdot 5 + 4$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 0$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 4 \neq 0$
0	0	1	1	0	0 because $(0011)_{2,4} = 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 3 = 0\cdot 5 + 3$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 0$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 3 \neq 0$
0	0	1	0	0	0 because $(0010)_{2,4} = 0\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 2 = 0\cdot 5 + 2$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 0$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 2 \neq 0$
0	0	0	1	0	0 because $(0001)_{2,4} = 0\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 1 = 0\cdot 5 + 1$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 0$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 1 \neq 0$
0	0	0	0	1	1 because $(0000)_{2,4} = 0\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 5 = 0\cdot 5 + 0$ so $(x_3x_2x_1x_0)_{2,4}$ **div** $5 = 0$ and $(x_3x_2x_1x_0)_{2,4}$ **mod** $5 = 0$

iii. $P_3 : \{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\} \to \{T, F\}$ given by

$$P_3(\ (x_3,x_2,x_1,x_0)\ ) = \begin{cases} T & \text{when } (x_3x_2x_1x_0)_{2,4} = [x_3x_2x_1x_0]_{s,4} \\ F & \text{otherwise} \end{cases}$$

Solution: The input-output table $\boxed{\text{does not}}$ implement P_3. We can see why not by considering the example (x_3,x_2,x_1,x_0) with $x_3 = 0$, $x_2 = 0$, $x_1 = 0$, $x_0 = 1$. Using the input-output definition table, the output for this example input is 0. However, using the definition of P_3, we get

$$P_3(\ (0,0,0,1)\ ) = 1$$

because

$$(0001)_{2,4} = 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1$$

and $[0001]_{s,4}$ has sign "+" and magnitude $0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 1$ so $(0001)_{2,4} = [0001]_{s,4}$. Since the output from the table doesn't match the predicate definition for this example, the input-output table and circuit do not implement P_3.

3. Recall the functions *mutation*, *insertion*, and *deletion* defined in class. We define the predicates:

Mut with domain $S \times S$ is defined by, for $s_1 \in S$ and $s_2 \in S$,

$$Mut(\ (s_1, s_2)\ ) = \exists k \in \mathbb{Z}^+ \exists b \in B(\ mutation(\ (s_1, k, b)\ ) = s_2\ )$$

Ins with domain $S \times S$ is defined by, for $s_1 \in S$ and $s_2 \in S$,

$$Ins(\ (s_1, s_2)\ ) = \exists k \in \mathbb{Z}^+ \exists b \in B(\ insertion(\ (s_1, k, b)\ ) = s_2\ )$$

Del with domain $\{s \in S \mid rnalen(s) > 1\} \times S$ is defined by, for $s_1 \in \{s \in S \mid rnalen(s) > 1\}$ and $s_2 \in S$,

$$Del(\ (s_1, s_2)\ ) = \exists k \in \mathbb{Z}^+(\ deletion(\ (s_1, k)\ ) = s_2\ )$$

For each quantified statement below, **first** translate to an English sentence.

Then, negate the **whole** statement and rewrite this negated statement so that negations appear only within predicates (that is, so that no negation is outside a quantifier or an expression involving logical connectives).

The translations are graded for fair effort completeness.

The negations are graded for correctness. For negations: You do not need to justify your work for this part of the question. However, if you include correct intermediate steps, we might be able to award partial credit for an incorrect answer.

Sample response that can be used as reference for the detail expected in your answer: Consider the statement

$$\forall n \in \mathbb{Z}^+\ \exists s \in S\ \ (\ L(\ (s, n)\ ) \wedge F_{\texttt{A}}(s) \wedge BC(\ (s, \texttt{A}, n)\ )\ )$$

Solution: English translation is

> For each positive integer there is some RNA strand of that length that starts with `A` and all of its bases are `A`.

We obtain the negation using multiple applications of De Morgan's rule and logical equivalences.

$$\begin{aligned}
&\neg\forall n \in \mathbb{Z}^+\ \exists s \in S\ \ (\ L(\ (s, n)\ ) \wedge F_{\texttt{A}}(s) \wedge BC(\ (s, \texttt{A}, n)\ )\ )\\
\equiv&\exists n \in \mathbb{Z}^+\ \neg\exists s \in S\ \ (\ L(\ (s, n)\ ) \wedge F_{\texttt{A}}(s) \wedge BC(\ (s, \texttt{A}, n)\ )\ )\\
\equiv&\exists n \in \mathbb{Z}^+\ \forall s \in S\ \neg(\ L(\ (s, n)\ ) \wedge F_{\texttt{A}}(s) \wedge BC(\ (s, \texttt{A}, n)\ )\ )\\
\equiv&\exists n \in \mathbb{Z}^+\ \forall s \in S\ \ (\ \neg L(\ (s, n)\ ) \vee \neg F_{\texttt{A}}(s) \vee \neg BC(\ (s, \texttt{A}, n)\ )\ )
\end{aligned}$$

(a) First statement:

$$\forall s \in S \ \ (\ Mut(\ (s,s) \ ) \leftrightarrow Ins(\ (s,s) \ ) \ )$$

Solution: English translation is

> For each RNA strand, there is a mutation that outputs the original strand if and only if there is an insertion that outputs the original strand.

We obtain the negation using multiple applications of De Morgan's rule and logical equivalences.

$$\begin{aligned}
&\neg \forall s \in S \ \ (\ Mut(\ (s,s) \ ) \leftrightarrow Ins(\ (s,s) \ ) \ ) \\
\equiv &\exists s \in S \ \neg (\ Mut(\ (s,s) \ ) \leftrightarrow Ins(\ (s,s) \ ) \ ) \\
\equiv &\exists s \in S \ \ (\ Mut(\ (s,s) \ ) \oplus Ins(\ (s,s) \ ) \ )
\end{aligned}$$

(b) Second statement

$$\forall s_1 \in S \ \forall s_2 \in S \ \forall s_3 \in S \ \ (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \to Mut(\ (s_1,s_3) \ ) \ )$$

Solution: English translation is

> For every choice of RNA strands s_1, s_2, s_3, if there is a mutation that transforms s_1 to s_2 and there is a mutation that transforms s_2 to s_3, then there is a mutation that transforms s_1 to s_3.

We obtain the negation using multiple applications of De Morgan's rule and logical equivalences.

$$\begin{aligned}
&\neg \forall s_1 \in S \ \forall s_2 \in S \ \forall s_3 \in S \ \ (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \to Mut(\ (s_1,s_3) \ ) \ ) \\
\equiv &\exists s_1 \in S \ \neg \forall s_2 \in S \ \forall s_3 \in S \ \ (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \to Mut(\ (s_1,s_3) \ ) \ ) \\
\equiv &\exists s_1 \in S \ \exists s_2 \in S \ \neg \forall s_3 \in S \ \ (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \to Mut(\ (s_1,s_3) \ ) \ ) \\
\equiv &\exists s_1 \in S \ \exists s_2 \in S \ \exists s_3 \in S \ \neg (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \to Mut(\ (s_1,s_3) \ ) \ ) \\
\equiv &\exists s_1 \in S \ \exists s_2 \in S \ \exists s_3 \in S \ \neg (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \to Mut(\ (s_1,s_3) \ ) \ ) \\
\equiv &\exists s_1 \in S \ \exists s_2 \in S \ \exists s_3 \in S \ \ (\ (\ Mut(\ (s_1,s_2) \ ) \wedge Mut(\ (s_2,s_3) \ ) \ ) \wedge \neg Mut(\ (s_1,s_3) \ ) \ )
\end{aligned}$$

(c) Third statement: we use S' to abbreviate $\{s \in S \mid rnalen(s) > 1\}$

$$\forall s_2 \in S' \ \exists s_1 \in S' \ \ (\ Del(\ (s_1,s_2) \ ) \ ) \wedge \neg \forall s_1 \in S' \ \exists s_2 \in S' \ \ (\ Del(\ (s_1,s_2) \ ) \ )$$

Solution: English translation is

> Each strand of length 2 or higher is the result of deletion from some strand of length 2 or higher and it's not the case that for each strand of length 2 or higher there is some strand of length 2 or higher that results from a deletion.

We obtain the negation using multiple applications of De Morgan's rule and logical equivalences.

$$\begin{aligned}
&\neg\,((\ \forall s_2 \in S'\ \exists s_1 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ )\ ) \land \neg\forall s_1 \in S'\ \exists s_2 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ )\ )\ ))\\
&\equiv (\neg\forall s_2 \in S'\ \exists s_1 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ )) \lor (\neg\neg\forall s_1 \in S'\ \exists s_2 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ ))\\
&\equiv (\exists s_2 \in S'\ \neg\exists s_1 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ )) \lor (\forall s_1 \in S'\ \exists s_2 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ ))\\
&\equiv (\exists s_2 \in S'\ \forall s_1 \in S'\ \ (\ \neg Del(\ (s_1, s_2)\ )\ )) \lor (\forall s_1 \in S'\ \exists s_2 \in S'\ \ (\ Del(\ (s_1, s_2)\ )\ ))
\end{aligned}$$

Bonus; not for credit (do not hand in): For each statement above, is the statement or its negation true? How do you know?