Sample Questions

Discrete Structures

(Midterm scheduled for Wednesday, February 24, 2021)

You must justify all your answers to recieve full credit

---

1 Boolean Expressions

Truth tables, logical equivalences, set operations, Venn diagrams.

1.(a). Given a statement in English and atomic propositions, write its Boolean expression.

1.(b). Given a Boolean expression, fill in missing values in its truth table.

1.(c). Given a Boolean expression equivalently transform it using Boolean identities.

1.(d). Given a Boolean expression, prove or disprove a tautology.

1.(e). Given a truth table, create a DNF or a CNF for it (and vice versa).

1.(f). Given a set expression, shade the regions in a Venn diagram that belong to it.

1.(g). Given two set expressions prove or disprove set identity or subset relation.

1.(a). Rewrite both English sentences (P_1 and P_2) as Boolean expressions using the given propositional variables.
P_1 := "If it is not snowing nor raining, then Gilbert has to tend his garden."
Propositional variables in P_1:
S := "It is snowing"
R := "It is raining"
G := "Gilbert has to tend his garden"
P_2 := "Jane will be a candidate in the elections regardless of whether she has a chance to be elected or not."
Propositional variables in P_2:
J := "Jane will be a candidate in the elections."
E := "She has a chance to be elected."

The construct "not A nor B" commonly means $\neg A \wedge \neg B$. The language construct "regardless" commonly means that all the conditions listed will imply the conclusion mentioned earlier. We rewrite the both sentences:

$$P_1 = \neg S \wedge \neg R \rightarrow G.$$
$$P_2 = (E \rightarrow J) \wedge (\neg E \rightarrow J).$$

Note. To get full credit for this, try to reflect the sentence structure as exactly as you can. Simplifying the expression P_1 as $\neg(S \vee R) \rightarrow G$ or writing simply $P_2 = J$ ("Jane will be candidate") is undesirable, since we want to reflect the logical structure of the English sentences; we do not want to apply any interpretations. □

1.(b). Consider the following Boolean expression $E = p \rightarrow q \rightarrow r$. Fill in the missing 4 slots in its truth table.

p	q	r	E
True	True	True	...
True	True	False	...
True	False	True	...
True	False	False	...
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	True

Rewrite the expression using the precedence/associativity rules for implication (and then plug in value $p =$ True, since all the values we need are in the upper part of the table, where p is true. Rewrite the expression:

$$\begin{aligned} & p \to q \to r \equiv \\ \equiv & \texttt{True} \to (q \to r) \equiv \\ \equiv & (q \to r). \end{aligned}$$

The only value False is when q is True, but r is False.

p	q	r	E
True	True	True	True
True	True	False	False
True	False	True	True
True	False	False	True
False	True	True	True
False	True	False	True
False	False	True	True
False	False	False	True

Note. In these transformations it is important to interpret the Boolean expression in the right way (using precedence/associativity rules when necessary). For example, if you wrongly interpret $E = p \to q \to r$ as $(p \to q) \to r$, you would get wrong truth table. □

1.(c). Transform the Boolean expression $p \vee q \wedge r$ into a logically equivalent one, using the same propositional letters p, q, r and two connectors: implication $\to$ and negaton $\neg$.

We know that $A \to B$ means $\neg A \vee B$ (for an implication to be True either its condition must be False, or the conclusion is true). Therefore

$$(A \vee B) \equiv (\neg A \to B). \tag{1}$$

Conjunction $\wedge$ has higher priority than $\vee$. So we transform the expression $p \vee (q \wedge r)$:

$$\begin{aligned} p \vee (q \wedge r) \equiv & \quad \text{Replace } \wedge \text{ by } \vee \text{ using De Morgan's law} \\ \equiv p \vee \neg(\neg q \vee \neg r) \equiv & \quad \text{Next, replace } \vee \text{ by } \rightarrow \text{ using identity (1)} \end{aligned}$$

□

1.(d). Prove or disprove that the following is a tautology:

$$(p \rightarrow q \rightarrow r) \leftrightarrow \neg(p \wedge q \wedge \neg r).$$

Rewrite the expression, replacing $A \rightarrow B$ as $(\neg A \vee B)$ and also the De Morgan's law:

$$\begin{aligned} & p \rightarrow (q \rightarrow r) \equiv \\ \equiv & \neg p \vee (q \rightarrow r) \equiv \\ \equiv & \neg p \vee (\neg q \vee r) \equiv \\ \equiv & \neg p \vee \neg q \vee r \equiv \\ \equiv & \neg(p \vee q \vee \neg r). \end{aligned}$$

We see that both expressions in $(\ldots) \leftrightarrow (\ldots)$ are indeed equivalent.

Note. Tautologies are typically proven using Boolean identities (like in the example above); building a full truth-table is usually too time-consuming and error-prone. If your question contains a non-tautology, the easiest way to disprove is with a counter-example: Just pick the values for p, q, r to make the value of the expression `False`. □

1.(e). Build the truth table for the following CNF:

$$f(A, B, C) = (A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C).$$

In the table below we insert value `False` in three places corresponding to the three clauses in the CNF. For example $(A \vee B \vee \neg C)$ in the CNF means that $f(\texttt{False}, \texttt{False}, \texttt{True}) = \texttt{False}$.

A	B	C	$f(A, B, C)$
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	True

□

1.(f). Let A, B, C be subsets in the same universe U. Draw a Venn diagram for these sets and shade the region corresponding to the set $S = A \oplus (B \oplus C)$.

Symmetric difference is commutative and associative, so we need to shade $S = A \oplus B \oplus C$ (it contains only those points which belong to odd number of sets A, B, C; i.e. either to one of these sets or to all three of them. See Figure 1.

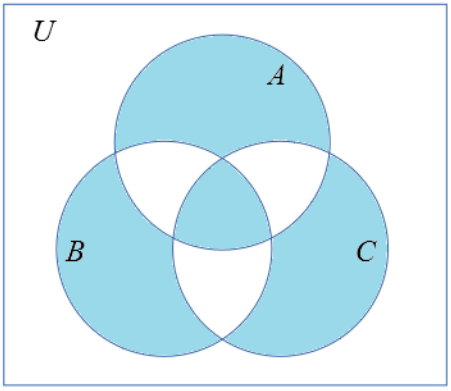

Figure 1: Venn diagram showing $A \oplus B \oplus C$.

□

1.(g). Let A and B be two arbitrary subsets of the same universe U. Prove or disprove the following set identity:

$$A \oplus (A \cap B) = A - B.$$

Method 1: Can use set identities like this:

$$\begin{aligned}
A \oplus (A \cap B) =& (A - (A \cap B)) \cup ((A \cap B) - A) = && \textit{(Definition of symmetric difference)} \\
=& (A - (A \cap B)) \cup \varnothing = && \textit{($A \cap B$ is a subset of A, so difference is $\varnothing$)} \\
=& (A \cap \overline{(A \cap B)}) = && \textit{(Replace set difference by an intersection)} \\
=& A \cap (\overline{A} \cup \overline{B}) = && \textit{(De Morgan's law: complement of an intersection)} \\
=& (A \cap \overline{A}) \cup (A \cap \overline{B}) = && \textit{(Distributive law)} \\
=& \varnothing \cup (A \cap \overline{B}) = A - B.
\end{aligned}$$

Method 2: Can use "element method": Assume that some $x \in (A \oplus (A \cap B))$ and prove that x must belong to $A - B$ and also in the opposite direction. (Try it yourself.)

Method 3: You can draw Venn diagrams for both sides of the equality. (Try it yourself.)

□

2 Quantifiers

Predicates, quantifiers, precedence, simple proofs.

2.(a). Given an English sentence and predicates, write its predicate expression.

2.(b). Given a predicate expression, restore parentheses, identify free/bound variables.

2.(c). Given a predicate expression, write its negation (De Morgan laws etc.).

2.(d). Given truth tables for predicates, evaluate nested quantifier expressions.

2.(e). Given a description of a set, define it in a set-builder notation.

2.(f). Given a pseudocode, write the predicate expression that it computes.

2.(a). Rewrite the statement as predicate expression.
Statement: "No judges are crooks; but there are judges who are elderly yet sharp-witted."
Domain: H is the set of all humans.
Predicates $J(x)$, $C(x)$, $E(x)$, $S(x)$ from H to $\{True, False\}$:
Predicate $J(x)$ is true iff human x is a judge.
Predicate $C(x)$ is true iff human x is a crook.
Predicate $E(x)$ is true iff human x is elderly.
Predicate $S(x)$ is true iff human x is sharp-witted.

The connective "but" is commonly understood as a conjunction (also "yet" means conjunction in this context). Also, the statement "no members of A are B" typically means that for all elements of A the proposition $\neg B$ must be true ($A(x) \to \neg B(x)$). Or, in other words, there is no such element of A such that B is also true.

So the statement consists of two subexpressions:

$$\forall x \in H(J(x) \to \neg C(x)) \wedge \exists x \in H(E(x) \wedge S(x)).$$

or equivalently:

$$\neg\exists x \in H(J(x) \wedge C(x)) \wedge \exists x \in H(E(x) \wedge S(x)).$$

Note. In the above expression, the scopes of two quantifiers do not interfere with each other (if you wish, you could also use different variable in both expressions).

$$\forall x \in H(J(x) \to \neg C(x)) \wedge \exists x \in H(E(x) \wedge S(x)).$$

Note. In this example it is important to read the meaning; it was stressed that the same judge can be elderly and sharp-witted. Writing $\exists x \in H\ E(x) \wedge \exists x \in H\ S(x)$ would distort the meaning (in this case elderly judges can be different people than the sharp-witted judges – even if they both exist). □

2.(b). Consider the following predicate expression:

$$\neg\exists y\ (\neg Q(x,z) \vee P(x,y) \wedge P(y,x)) \wedge \forall x\ (Q(y,z) \to \neg P(x,y) \to P(y,x)). \qquad (2)$$

Rewrite the equation (2); insert all the parentheses so that **every** subexpression serving as an argument for Boolean operations and quantifiers is in parentheses. Use the rules for precedence for Boolean operators:
Rule1: Decreasing order of precedence: $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$
Rule2: Both quantifiers ($\forall$, $\exists$) have the same (highest) precedence as $\neg$. **Rule3:** Implication and equivalence are right-associative; conjunction and disjunction are left-associative.

Also underline those variables which are bound. Leave all the unbound variables without underlining.

Negations have the highest priority, surround them by parentheses:

$$\neg\exists y\ ((\neg Q(x,z)) \vee P(x,y) \wedge P(y,x)) \wedge \forall x\ (Q(y,z) \to (\neg P(x,y)) \to P(y,x)).$$

Conjunctions have the next highest priority, surround them by parentheses:

$$\neg\exists y\ ((\neg Q(x,z)) \vee (P(x,y) \wedge P(y,x))) \wedge \forall x\ (Q(y,z) \to (\neg P(x,y)) \to P(y,x)).$$

Implication is right associative, so we group it from the right:

$$\neg\exists y\ ((\neg Q(x,z)) \vee (P(x,y) \wedge P(y,x))) \wedge \forall x\ (Q(y,z) \to ((\neg P(x,y)) \to P(y,x))).$$

Finally, surround in parentheses the larger subexpresisons:

$$(\neg(\exists y\ ((\neg Q(x,z)) \vee (P(x,y) \wedge P(y,x))))) \wedge (\forall x\ (Q(y,z) \to ((\neg P(x,y)) \to P(y,x)))).$$

Underline the bound variables (optionally, highlight the scope for every quantifier):

$$(\neg(\exists \underline{y}\ ((\neg Q(x,z)) \vee (P(x,\underline{y}) \wedge P(\underline{y},x))))) \wedge (\forall \underline{x}\ (Q(y,z) \to ((\neg P(\underline{x},y)) \to P(y,\underline{x})))).$$

In particular, the quantifier $\forall x$ only binds the variables x in the second subexpression; the blue x and the underlined $\underline{x}$ have nothing in common (the bound variables could be renamed without any effect on the meaning of the expression). □

2.(c). Simplify the expression with negation so that negation is only applied to individual predicates or propositional variables (rather than larger subexpressions or quantifiers):

$$\neg\,(\exists x\forall y\,(P(x,y) \to Q(x,y)) \vee \exists y\forall x(\neg P(x,y) \wedge Q(x,y)))$$

$$\begin{aligned}
&\neg\,(\exists x\forall y\,(P(x,y) \to Q(x,y)) \vee \exists y\forall x(\neg P(x,y) \wedge Q(x,y))) \equiv\\
\equiv&\neg\,(\exists x\forall y\,(P(x,y) \to Q(x,y))) \wedge \neg\,(\exists y\forall x(\neg P(x,y) \wedge Q(x,y))) \equiv\\
\equiv&(\forall x\neg\forall y\,(P(x,y) \to Q(x,y))) \wedge (\forall y\neg\forall x(\neg P(x,y) \wedge Q(x,y))) \equiv\\
\equiv&(\forall x\exists y\neg\,(\neg P(x,y) \vee Q(x,y))) \wedge (\forall y\exists x\neg(\neg P(x,y) \wedge Q(x,y))) \equiv\\
\equiv&(\forall x\exists y\,(\neg\neg P(x,y) \wedge \neg Q(x,y))) \wedge (\forall y\exists x(\neg\neg P(x,y) \vee \neg Q(x,y))) \equiv\\
\equiv&(\forall x\exists y\,(P(x,y) \wedge \neg Q(x,y))) \wedge (\forall y\exists x(P(x,y) \vee \neg Q(x,y)))\,.
\end{aligned}$$

Note. Normally you would simplify the negation expression much faster without showing all the steps. But in your answer (like any other algebraic transformation), make sure to show at least 1–2 intermediate steps, not the bare answer. This also allows to get partial credit, if you have some typos in your answer. □

2.(d). Check the following nested predicate statement on the predicates defined below:

$$\forall x \in \mathbf{N}\ \exists y \in \mathbf{N}\ ((y > x) \wedge \neg(P(y) \to Q(y))). \tag{3}$$

$P(x)$ defined on $\mathbf{N} = \{0, 1, 2, \ldots\}$; it is `True` iff x is full square.
$Q(x)$ defined on $\mathbf{N}$; it is true iff the last digit of x is not 1 or 2.

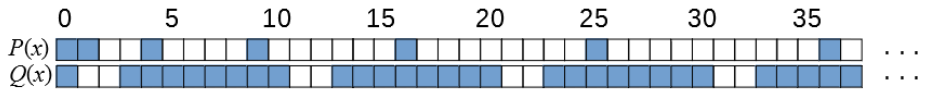

Figure 2: Predicate values that equal to `True` are shaded; `False` are white.

Determine whether the predicate expression (3) is `True` or `False`; explain your answer.

Both predicates are defined on the infinite set $\mathbf{N} = \{0, 1, 2, 3, \ldots\}$ of natural numbers. Some initial values are shown in Figure 2.

The predicate subexpression $\forall x \in \mathbf{N}\ \exists y \in \mathbf{N}\ ((y > x) \wedge (\ldots))$ can be read as follows: "There are arbitrarily large natural numbers y such that ...."
Literally: For any (no matter how large) $x \in \mathbf{N}$ there is an even larger $y > x$ satisfying the subexpression ....

In our case, the subexpression in place of $(\ldots)$ is $\neg(P(y) \to Q(y))$. Namely, we want to find y such that $P(y)$ is `True` (y is a full square), but $Q(y)$ is `False` (the last digit is 1 or 2).

It is easy to see that there are indeed arbitrarily large numbers y such that their square ends with digit 1. (For example, $y = 9^2 = 81$, $y = 19^2 = 361$, $y = 29^2 = 841$, etc. are all valid answers; they are all full squares and end with digit 1. For any x you can find an even bigger $y = (10k + 9)^2$ with this property. □

2.(e). The following expression:

$$\{x \in U \mid P(x)$$

is the regular set-builder notation: It denotes a subset of the universe U consisting of all those x that make the predicate $P(x)$ true.

Use this set-builder notation to describe the set of all full squares that have the last digit equal to 6 (namely, the set $\{16, 36, 196, 256, \ldots\}$). You can use the set of all integers $\mathbf{Z}$ as the universe, the arithmetic operations (including $(a \bmod b)$, the remainder dividing a by b), Boolean operations and quantifiers.

Full squares are numbers for which there exists a number m such that $m^2 = x$.
Moreover, a number ends with digit 6, if its remainder when divided by 10 equals 6.
We use these facts to create the set-builder notation:

$$\left\{x \in \mathbf{Z} \mid \exists m \in \mathbf{Z}\ \left(m^2 = x \wedge (m^2 \bmod 10) = 6\right)\right\}.$$

□

2.(f). Assume that there are two lists $A[1..100]$ and $B[1..100]$ containing 100 integer numbers each. They are passed to the function computing MyPredicate(A, B) on these lists (see pseudocode below). This function returns value `True` or `False` depending on the numbers in lists A and B.

```
    MyPredicate(A : IntegerList, B : IntegerList)
1   val := True
2   for i := 1 to 100
3       if A[i] > 0
4           if (not B[i] > 0)
5               val := False
6   return val
```

Write the predicate expression that is computed by this function. The expression can use quantifiers $\forall i \in \{1, \ldots, 100\}$ and $\exists i \in \{1, \ldots, 100\}$; all Boolean connectors, references to array elements $A[i]$ and $B[i]$ as well as equality and inequality predicates.

The function MyPredicate(A, B) starts with the result $val =$ `True` (but it can later switch to `False`; and never can flip back to `True`). Therefore it is a $\forall$-type quantifier (it is true for the empty set and also those sets, where the **if** statement on Line 4 never happens).

The condition that makes this predicate false is when $A[i] > 0$, but $B[i] \leq 0$. Therefore the predicate expression computed by this function is this:

$$\forall i \in \{1, \ldots, 100\} \;\; (A[i] > 0 \rightarrow B[i] > 0)\,.$$

□