Section 1.7: The Algebra of Sets

Membership Tables

Let's consider the sets A and B. As we learned previously, if we want to be able to talk about the negation of these sets, $\overline{A}$ and $\overline{B}$, then we also have to specify the universal set U. So let us now examine an element x which is taken from U. There are four cases:

1: $x \notin A$ and $x \notin B$
2: $x \notin A$ and $x \in B$
3: $x \in A$ and $x \notin B$
4: $x \in A$ and $x \in B$

If we want to find out under which conditions x belongs to $A \cup B$ or $A \cap B$, we can draw what's called a membership table:

$x \in A$	$x \in B$	$x \in (A \cup B)$	$x \in (A \cap B)$
no	no	no	no
no	yes	yes	no
yes	no	yes	no
yes	yes	yes	yes

More often, we abbreviate the "no" and "yes" to 0 and 1, respectively, and drop the "$x \in$" notation to get a slightly shorter membership table:

A	B	$A \cup B$	$A \cap B$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

If we wish to write out a membership table for $A \cup \varnothing$, we must remember that our element x cannot belong to $\varnothing$, so the table would look like the following.

A	$\varnothing$	$A \cup \varnothing$
0	0	0
1	0	1

We see from this table that $A \cup \varnothing$ is equivalent to A, or $A \cup \varnothing = A$, as we found in the section on operations on sets.

Similarly, if we wish to examine $A \cap U$, then we have to remember that *x* will always belong to U, so

A	U	$A \cap U$
0	1	0
1	1	1

And, as before, we see that $A \cap U = A$. (Finding $A \cup U$ and $A \cap \varnothing$ has been left for the exercises.)

Example:

Write out the membership table for $A \cup \overline{A}$. What set does this represent?

Answer: The membership table is:

A	$\overline{A}$	$A \cup \overline{A}$
0	1	1
1	0	1

and since the final column is all 1s, then $A \cup \overline{A} = U$, as we have found before.

Example:

Write out the membership table for $\overline{A} \cap (B \cup \overline{C})$.

Answer:

A	B	C	$\overline{C}$	$B \cup \overline{C}$	$\overline{A}$	$\overline{A} \cap (B \cup \overline{C})$
0	0	0	1	1	1	1
0	0	1	0	0	1	0
0	1	0	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0

Computer Representation of Sets

How might we represent sets on a computer? Consider the sets U = {1, 2, 3, … 8} with A = {1, 3, 5, 7}, B = {2, 3} and C = {1, 2, 3, 4, 6, 7, 8}. Then we can represent the sets A, B, and C as **bit strings** (series of 0s and 1s) as follows. We write a table with the elements listed across the top and each row corresponds to a single set. Then for the set A, if each element belongs to A, then that cell gets a 1. Otherwise, it gets a 0. So for set A, you get ones in the 1, 3, 5, and 7 columns and zeroes in the others.

element	1	2	3	4	5	6	7	8
U	1	1	1	1	1	1	1	1
A	1	0	1	0	1	0	1	0
B	0	1	1	0	0	0	0	0
C	1	1	1	1	0	1	1	1

So A = [10101010] and $\overline{A}$ is then just [01010101]. $\overline{A} \cap B$ can be found by writing the bit strings in a table. Take each column and intersect (AND) each pair of ones and zeros to give

$\overline{A}$	0	1	0	1	0	1	0	1
B	0	1	1	0	0	0	0	0
$\overline{A} \cap B$	0	1	0	0	0	0	0	0

which means that $\overline{A} \cap B$ = [01000000], which corresponds to the set {2}.

Example:

Take the sets above and find $\overline{C}$, $\overline{A} \cup \overline{C}$, and $\overline{A \cap C}$.

Answer: $\overline{C}$ = [00001000] = {5}. (Be sure to translate your answer back into proper set notation!)

A	1	0	1	0	1	0	1	0
$\overline{A}$	0	1	0	1	0	1	0	1
C	1	1	1	1	0	1	1	1
$\overline{C}$	0	0	0	0	1	0	0	0
$\overline{A} \cup \overline{C}$	0	1	0	1	1	1	0	1
$A \cap C$	1	0	1	0	0	0	1	0
$\overline{A \cap C}$	0	1	0	1	1	1	0	1

So $\overline{A} \cup \overline{C}$ = [01011101] = {2, 4, 5, 6, 8}. You'll notice that this is the same result as $\overline{A \cap C}$, which is another example of De Morgan's law.

This whole procedure, the computer representation of sets, may seem a lot of work for an operation that you could do by looking at the original sets. However, imagine that the universal set had 100,000 elements and each set A, B, and C contained at least 10,000 elements. Letting a computer translate each set to bit strings and then do all the hard work for you might then sound like a good plan!