ITM 207: Simplifying Expressions

PROPERTIES OF BOOLEAN ALGEBRA, SIMPLIFICATION TIPS

Note: Please review the boolean operators or refer to the Boolean Basics tip sheet prior to viewing this tip sheet.

Properties of Boolean Algebra

PROPERTY	AND	OR
Commutative	AB = BA	A + B = B + A
Associative	(AB)C = A(BC)	(A + B) + C = A + (B + C)
Distributive	A(B + C) = (AB) + (AC)	A + (BC) = (A + B) (A + C)
Identity	A1 = A	A + 0 = A
Complement	A(A′) = 0	A + (A′) = 1
De Morgan's law	(AB)′ = A′ OR B′	(A + B)′ = A′B′

Before we begin to simplify expressions, we must first take a look at the properties of boolean algebra listed in the table above. Some of these properties work the same way in regular algebra, so we will put more effort in exploring the ones more uniquely applicable to boolean algebra:

Distributive AND:
While this property is the same as distributing and factoring out a common term in regular algebra, it will be used extensively in boolean algebra to either isolate or distribute a variable. As we will see later in the examples, it is great for setting up the expression for the use of the **Complementary OR**.

Distributive OR:
This distributive property would not work in regular algebra. It only functions in boolean algebra due to the two-valued nature of the boolean data type. The proof can be obtained by expanding and simplifying the right side. Like the Distributive AND, it is used to isolate and distribute variables to set up for further simplifications.

Complement AND:
Since a boolean variable can only be either 1 or 0, any variable multiplied by its complement will result in either 1·0 or 0·1, both of which will equal 0. This property is a crucial part of boolean algebra simplification since it cancels out compliments and the resulting 0 can further reduce other parts of the expression.

Complement OR:
Similar to the **Complement AND**, if a boolean variable can only be either 1 or 0, any variable summed with its complement will result in either 1 + 0 or 0 + 1, both of which will equal 1. This property is just as important since it also cancels out complements. The resulting 1 can either be reduced with the **Identity AND** or be used to reduce other terms connected by addition.

De Morgan's Law AND:
This property simplifies **NAND** gate expressions, breaking variables out of the bracket for further simplification. Recall the behaviour of NAND gates:

- The output will be 0 if all input signals are 1, otherwise it will be 1

Another way to interpret it is:

- As long as one input signal is 0, the output will be 1

Therefore, for (AB)′ to be 1, either A is not 1 or B is not 1, thus the expression of A′ + B′. For those who have taken SSH105, consider the following interpretation:

- For the statement "It is not True that both A and B are true" to be true, either A is not true (A′) or B is not true (B′)

De Morgan's Law OR:
This property simplifies **NOR** gate expressions, breaking variables out of the bracket for further simplification. Recall the behaviour of NOR gates:

- The output will be 0 as long as one input is 1, otherwise it will be 1

Another way to interpret it is:

- The output will be 1 only if all inputs are 0, otherwise it will be 0

Therefore, for (A + B)′ to be 1, both A and B must be 0, thus the expression of A′B′. For those who have taken SSH105, consider the following interpretation:

- For the statement "Neither A or B is true" to be true, both A and B must be false (A′ and B′)

Tip: The explanation of these properties are for your better understanding. It may be easier to memorize the properties (especially De Morgan's Law) and write them down immediately after the test starts. This way, you can quickly refer back without having to recall the reasoning behind them.

Simplification Tips:

An expression can be simplified in different ways, so there are no specific steps to follow. Instead, here are some things to look out for that can help guide you through the process:

Look for like terms and factorize:

While an expression may seem like it cannot be operated on, factoring it using the Distributive AND can open up different angles to continue the simplification.

- Example: AB + AC + AB′

It may seem like none of the properties can be immediately used on this expression, but we can factor it to give it a new look:

• **A**B + **A**C + **A**B′	- Identify like terms
• **A**(B + C + B′)	- Factor it out using Distributive AND

Now if we refer back to the properties, we may see more applicable ones..

• A(B + **C + B′**) A(B + **B′ + C**)	- Switch B′ and C with the Commutative OR
• A(**B + B′** + C) A(**1** + C)	- Reduce with Complement OR
• A(**1**)	- 1 plus anything will yield 1 in boolean
• A	- Identity AND

Look for compliments:

Why did we switch the position of B′ and C with the Commutative OR in the previous example? It was to put the B′ next to B so we could use the Complement OR. The switch was not a necessary step, but it illustrates the importance of **matching compliments**.

The goal of simplifying boolean expressions is to reduce them to their shortest forms. The complement properties are crucial because they are what cancels out components to shorten expressions. If we look at things this way, simplifying expressions becomes a matching puzzle game. We take what looks similar, put them next to each other, and they disappear with a poof.

If we focus on finding and matching complements, we could have approached the previous example differently and still reach the same answer:

• A**B** + AC + A**B′**	- Identify complements and look for ways of matching them together
• AB + **AC + AB′** AB + **AB′ + AC**	- Commutative OR
• A(**B + B′**) + AC	- Factor out A from the first two terms using Distributive AND to isolate the complement pair
• A(**1**) + AC	- Complement OR
• A + AC	- Identity AND
• **A** + **A**C **A**(1 + C)	- Factor out A again
• A(1)	- 1 plus anything equals 1
• A	- Identity AND

<u>Look for structural similarity and refer back to properties:</u>
The better you know your properties, the quicker you will be able to spot where to apply them in an expression. Keeping the table of boolean algebra properties by your side while you solve practice questions will help you become more familiar with the properties and finish those questions with better efficiency.

The table shows the most basic formats of the properties. Although in real questions, they will show up quite differently:

- Example: (A′(BC + C′))′

This example may look intimidating at first, but if we use separation of concerns and try to match its characteristics with the properties, we can break it down into more digestible chunks:

• (A′(B**C** + **C′**))′	- Identified complements
• (A′(**BC + C′**))′	- Structure of the Distributive OR
• **(**A′(BC + C′)**)′**	- Structure of De Morgan’s Law AND, with A′ as one term inside the bracket and (BC + C′) as the other

And with that, let's pick a component and start simplifying:

- $\mathbf{(}A'(BC + C')\mathbf{)'}$
$(A')' + (BC + C')'$ — Let's start with the De Morgan's Law AND to get rid of some brackets so the expression looks less complex; here is how we can apply the similar structure:

  Property: $(AB)' = A' + B'$
Question: $(A'(BC + C'))' = (A')' + (BC + C')'$
- $\mathbf{(A')'} + (BC + C')'$
$\mathbf{A} + (BC + C')'$ — The complements here cancel each other out
- $A + \mathbf{(BC + C')'}$
$A + \mathbf{(BC)'(C')'}$ — This part has the same structure as the De Morgan's Law OR
- $A + (BC)'\mathbf{(C')'}$
$A + (BC)'C$ — Again, the complements cancel out
- $A + \mathbf{(BC)'}C$
$A + \mathbf{(B' + C')}C$ — De Morgan's Law AND
- $A + \mathbf{C}(B' + C')$ — Commutative AND
- $A + \mathbf{C}B' + \mathbf{C}C'$ — Distributive AND
- $A + CB' + \mathbf{CC'}$
$A + CB' + \mathbf{0}$ — Complement AND
- $A + B'C$ — Commutative AND for some optional alphabetical formatting and we're done!

Tip: An easy way to check your answers or get unstuck from a practice question is to use an online boolean algebra calculator. Though make sure to check the step-by-step solution so you learn from the process and not become reliant on the calculator. It won't be there to help you out during an exam!