Simplifying Boolean Expressions

Learn how to simplify complex boolean expressions!

Simplifying Boolean Expressions

As our boolean expressions continue to grow in complexity, we can use different methods to help us rewrite and simplify complex expressions without changing its final value.

When would rewriting boolean expression become useful? After all, our programs solve the equation for us. We should know how to simplify boolean expressions for situations when we don't have a computer in front of us. If we're left to solve these expressions on our own (like during the AP Computer Science A Exam), knowing how to simplify complicated expressions can come in handy.

Using methods like DeMorgan's Laws and changing operators to their opposite value can allow us to take a complex expression like:

```
!(a > b && a == b)
```

and rewrite it as:

```
a <= b || a != b
```

The two expressions above are equivalent; this means that, no matter what the value of `a` or `b` is, both expressions will return the same result.

Applying DeMorgan's Laws

DeMorgan's laws are a set of rules we can apply to boolean expressions in order to transform the expression without changing its overall value.

DeMorgan's first law explains that two expressions that are negated together (using `!`) and compared with AND (`&&`) are equivalent to two separately negated expressions compared with OR (`||`):

```
!(exp1 && exp2) == !(exp1) || !(exp2)
```

The second law proves that, on the opposite hand, two expressions that are negated together and compared with OR are equivalent to two individually negated expressions compared with AND:

```
!(exp1 || exp2) == !(exp1) && !(exp2)
```

Going back to our above example, let's break down how we can use DeMorgan's laws to simplify this expression:

```
!(a > b && a == b)
```

Using the information from the first law, we'll move `a > b` and `a == b` into separate parentheses prepended with `!`. Then, we'll change `&&` to `||` which will give us the following expression:

```
!(a > b) || !(a == b)
```

Removing Negation with Opposite Operators

We can eliminate NOT operators (`!`) from our expressions by changing all operators to their negated, or opposite, value.

Let's continue to simplify the expression `!(a > b) || !(a == b)` by rewriting it so that the expression doesn't use `!`. We'll start with the operand on the right: `!(a > b)`. In order to remove `!` without changing the expression's value, we must change the `>` operator to its opposite value.

We can use the following table to find an operator's negated value:

Operator	Negated Operator
`==`	`!=`
`!=`	`==`
`>`	`<=`
`<`	`>=`
`<=`	`>`
`>=`	`<`

With this information, we know `>` will become `<=`, making our expression `!(a > b)` become `a <= b`. Now, we'll convert the other part of the expression `!(a == b)` to `a != b`.

Putting it all together, our final expression looks like this:

```
a <= b || a != b
```

Despite the expressions `!(a > b && a == b)` and `a <= b || a != b` looking different, they are equivalent boolean values. No matter what values we give `a` and `b`, the two expressions will result in the same value.

Review

Great job reaching the end of this article. We now have in our pockets methods for simplifying boolean expressions. Let's review what we learned:

- DeMorgan's Laws can be used to rewrite expressions.
  - The first law states that two expressions that are negated together and compared using `&&` is equivalent to two separately negated expressions compared with `||`.
  - The second law states that two expressions that are compared with `||` and are negated together are equivalent to two separately negated expressions compared with `&&`.
- To remove the NOT operator from an expression, we can replace an operator with its opposite operator value.
- A boolean expression created using these above methods are equivalent to their original expression. They will always produce the same truth value.

Code Challenge

Now it's your turn to simplify a `boolean` expression. Take a look at the expression placed in `exp1`.

Using only DeMorgan's laws, simplify the expression in `exp1` and place your solution in a `boolean` variable called `exp2`.

Then, continue to simplify the expression by using negated operators to remove the `!` operators from `exp2`. Place this simplified expression in a `boolean` called `exp3`.

Finally, print the values of `exp1`, `exp2`, and `exp3`. Do they have the same output? Does the output remain the same if the values of `a` and `b` are changed?

```
1  public class Simplify {
2    public static void main(String[] args) {
3
4      int a = 5;
5      int b = 7;
6
7      boolean exp1 = !(a == b && b >= a);
8
9      // Add your code below
10
11
12
13   }
14 }
```